cppNum Self-Study Course

Exercises

Uwe Naumann

Informatik 12: Software and Tools for Computational Engineering (STCE)
RWTH Aachen University

1. Iterative Solver for Indefinite Linear Systems

- Extend cppNum v2.1 with a suitable iterative solver for indefinite linear systems and use it for the solution of systems of nonlinear equations by the Newton method.
- Design at least three scalable (in the dimension of the nonlinear system) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solution produced by LU factorization.
- Document your analysis, design, implementation, case studies and project management.

2. Iterative Solver for spd Linear Systems

- Extend cppNum v2.4 with a suitable iterative solver for symmetric positive definite (spd) linear systems and use it for the solution of convex optimization problems by the Newton method.
- Design at least three scalable (in the dimension of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solution produced by LL^T factorization.
- Document your analysis, design, implementation, case studies and project management.

3. Finite Difference Approximation of all Derivatives

- Extend cppNum v2.4. and v2.5 with methods for approximating all derivatives required by finite differences.
- Design at least three scalable (in the dimensions of the objective function, resp. of the residual of the system of ordinary differential equations) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by algorithmic differentiation.
- Document your analysis, design, implementation, case studies and project management.

4. Algorithmic Differentiation with ADOL-C

- Modify cppNum v2.4. and v2.5 such that all derivatives are computed with ADOL-C (https://github.com/coin-or/ADOL-C)
- Design at least three scalable (in the dimensions of the objective function, resp. of the residual of the system of ordinary differential equations) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the built-in algorithmic differentiation solution.
- Document your analysis, design, implementation, case studies and project management.

5. Runge-Kutta Scheme

- Extend cppNum v2.5 with a Runge-Kutta scheme for the solution of initial value problems for systems of ordinary differential equations.
- Design at least three scalable (in the dimensions of the residual of the system of ordinary differential equations) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the explicit and implicit Euler methods.
- Document your analysis, design, implementation, case studies and project management.

6. Conjugate Gradient Minimizer

- Extend cppNum v2.3 with a conjugate gradient minimizer for convex nonlinear objective functions.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods.
- Document your analysis, design, implementation, case studies and project management.

7. BFGS Minimizer

- Extend cppNum v2.3 with a BFGS minimizer for convex nonlinear objective functions.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods.
- Document your analysis, design, implementation, case studies and project management.

8. SR1 Minimizer

- Extend cppNum v2.3 with an SR1 minimizer for convex nonlinear objective functions.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods.
- Document your analysis, design, implementation, case studies and project management.

9. Random Multistart with Gradient Descent

- Extend cppNum v2.3 with a random multistart method for minimizing nonlinear objective functions with several local minima. Use uniform random sampling over the free domain.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods using a single sample point.
- Document your analysis, design, implementation, case studies and project management.

10. Random Multistart with Newton Method

- Extend cppNum v2.4 with a random multistart method for minimizing nonlinear objective functions with several local minima. Use uniform random sampling over the free domain.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods using a single sample point.
- Document your analysis, design, implementation, case studies and project management.

11. Linear Regression

- Extend cppNum v2.4 with the normal equation method for solving linear regression problems.
- Design at least three scalable (in the dimensions of the objective function) sufficiently distinct case studies for run time experiments.
- Compare numerical results and run times with the solutions obtained by the gradient descent and Newton methods using a single sample point.
- Document your analysis, design, implementation, case studies and project management.

12. Exception Handling

- Extend cppNum v2.4 and v2.5 with appropriate C++ exception handling.
- Design at least three scalable (in the dimensions of the objective function, resp. of the residual of the system of ordinary differential equations) sufficiently distinct case studies for run time experiments.
- Compare general behavior and run times with the exception handling-free versions.
- Document your analysis, design, implementation, case studies and project management.

13. Checkpointing for "Cold" Restarts

- Extend cppNum v2.5 with a checkpointing mechanism allowing for "cold" restarts from designated intermediate states of the iterative implicit Euler integration method.
- Design at least three scalable (in the dimension of the residual of the system of ordinary differential equations) sufficiently distinct case studies for run time experiments.
- Compare general behavior and run times with the checkpointing-free versions (requiring restarts "from scratch" in case of faulty behavior).
- Document your analysis, design, implementation, case studies and project management.