Software Lab Computational Engineering Science

Tutorial Exercises

Uwe Naumann

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

RWTH Aachen University

Contents

Outline

- Extend LINEAR_SYSTEM with an iterative (e.g, conjugate gradients) solver and use it in NONLINEAR_SYSTEM for the solution of the Newton system.
- Design at least one scalable (in the dimension of the nonlinear system) case study for run time experiments.
- Compare numerical results and run times with the original solution (direct linear solvers).
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

Outline

Sparse Eigen Data Structures and Linear Solvers

- Replace Eigen/Dense by Eigen/Sparse[1] in `LINEAR_SYSTEM`.
- Design at least one scalable (both in the size of the state $\mathbf{x}$ and in the number of parameters in $\mathbf{p}$) case study for run time experiments in the context of the `NONLINEAR_SYSTEM` library.
- Compare numerical results and run times with dense version.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

[1] `eigen.tuxfamily.org/dox/group__Sparse__chapter.html`

Outline

Tutorial Exercise 3

Replacement of dco/c++ by ADOL-C

- Replace dco/c++ by ADOL-C[2] in `NONLINEAR_SYSTEM`. Use ADOL-C both for the computation of the Jacobian of the residual within Newton's method and for parameter sensitivity analysis similar to toy_dco.cpp.
- Design at least one scalable (both in the size of the state $\mathbf{x}$ and in the number of parameters in $\mathbf{p}$) case study for run time experiments.
- Compare numerical results and run times with the dco/c++ version.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

[2] `github.com/coin-or/ADOL-C`

Outline

- Extend the implicit solver in `ODE_SYSTEM` such that parameterized systems of explicit linear ordinary differential equations are solved by a linear (instead of the Newton) solver.
- Allow for use of LL^T or LDL^T factorization in case of known symmetry of the Jacobian of the residual.
- Design at least one scalable (in the dimension of the system) case study for run time experiments.
- Compare numerical results and run times with the original (suboptimal) solution.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

Outline

Tangent Sensitivity Analysis of ODEs

- Design a scalable (in the number of free parameters, i.e, in the size of $\mathbf{p}$) case study for `ODE_SYSTEM`; exchange with group working on Tutorial Exercise 6.
- Use dco/c++ and ADOL-C for parameter sensitivity analysis of the L_2-norm of the final state $\|\mathbf{x}\|_2$ of the system, i.e. for the computation of the gradient of $\|\mathbf{x}\|_2$ with respect to $\mathbf{p}$.
- Apply both dco/c++ and ADOL-C in tangent mode. Compare numerical results and run times (scaling); exchange with group working on Tutorial Exercise 6.
- Document your
  - analysis
  - design
  - implementation
  - case study

  with a slide deck similar to the one used for the lecture.

Outline

- Design a scalable (in the number of free parameters, i.e, in the size of $\mathbf{p}$) case study for `ODE_SYSTEM`; exchange with group working on Tutorial Exercise 5.
- Use dco/c++ and ADOL-C for parameter sensitivity analysis of the L_2-norm of the final state $\|\mathbf{x}\|_2$ of the system, i.e. for the computation of the gradient of $\|\mathbf{x}\|_2$ with respect to $\mathbf{p}$.
- Apply both dco/c++ and ADOL-C in adjoint mode. Compare numerical results and run times (scaling); exchange with group working on Tutorial Exercise 6.
- Document your
  - analysis
  - design
  - implementation
  - case study

  with a slide deck similar to the one used for the lecture.

Outline

- Iterative Linear Solver
- Sparse Eigen Data Structures and Linear Solvers
- Replacement of dco/c++ by ADOL-C
- Linear Solvers for Linear ODEs
- Tangent Sensitivity Analysis of ODEs
- Adjoint Sensitivity Analysis of ODEs
- Runge-Kutta Schemes for ODEs
- Optimization of Parameterized ODEs
- Calibration of Parameterized Nonlinear Systems
- Convex Unconstrained Optimization

- Extend `ODE_SYSTEM` by at least two explicit Runge-Kutta methods.
- Design at least one scalable (in the size of the state $\mathbf{x}$) case study for `ODE_SYSTEM` for run time experiments.
- Compare numerical results and run times with those computed by the given (explicit and implicit) Euler methods
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

Outline

Iterative Linear Solver

Sparse Eigen Data Structures and Linear Solvers

Replacement of dco/c++ by ADOL-C

Linear Solvers for Linear ODEs

Tangent Sensitivity Analysis of ODEs

Adjoint Sensitivity Analysis of ODEs

Runge-Kutta Schemes for ODEs

Optimization of Parameterized ODEs

Calibration of Parameterized Nonlinear Systems

Convex Unconstrained Optimization

- Extend `ODE_SYSTEM` by a simple gradient descent method for minimizing the L_2-norm of the final state $\|\mathbf{x}\|_2$ of the system as a function of the free parameters in $\mathbf{p}$.
- Use dco/c++ for the computation of the gradient.
- Design at least one scalable (in the number of free parameters, i.e, in the size of $\mathbf{p}$) case study for run time experiments.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

Outline

- Extend NONLINEAR_SYSTEM by a simple gradient descent method for fitting the final state $\mathbf{x}$ of the system to "observations" (perturbed simulations) $\tilde{\mathbf{x}}$ through calibration of the free parameters $\mathbf{p}$. Minimize the least-squares objective $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2$.
- Use dco/c++ for the computation of the gradient of the objective with respect to $\mathbf{p}$.
- Design at least one scalable (in the number of free parameters, i.e, in the size of $\mathbf{p}$) case study for run time experiments.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.

Outline

- Implement a convex unconstrained optimization method by application of `NONLINEAR_SYSTEM` to the necessary optimality criterion $\frac{df}{d\mathbf{x}}(\mathbf{x}(\mathbf{p}), \mathbf{p}) = 0 \in R^n$ for parameterized objective functions $f = f(\mathbf{x}(\mathbf{p}), \mathbf{p})$ to be specified by the end user.
- Use dco/c++ in adjoint mode for the computation of the gradient and of the Hessian of the objective with respect to $\mathbf{x}$.
- Use Eigen for checking the sufficient optimality condition (Hessian $\frac{d^2 f}{d\mathbf{x}^2}(\mathbf{x}(\mathbf{p}), \mathbf{p}) \in R^{n \times n}$ should be positive definite at the stationary point $\mathbf{x}^*$).
- Design at least one scalable (in the number of free parameters, i.e, in the size of $\mathbf{p}$) case study for run time experiments.
- Document your
  - analysis
  - design
  - implementation
  - case studies

  with a slide deck similar to the one used for the lecture.