Hyperelasticity based Mesh-Deformation

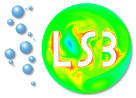
Malte Schuh

Januar 29th, 2019

technische universität
dortmund

fakultät für
mathematik

Contents



Motivation

Mesh-deformation is useful for

- complex geometries
- multi-phase flow
- FSI
- ...

At the moment: Only the outer boundary changes.

Task

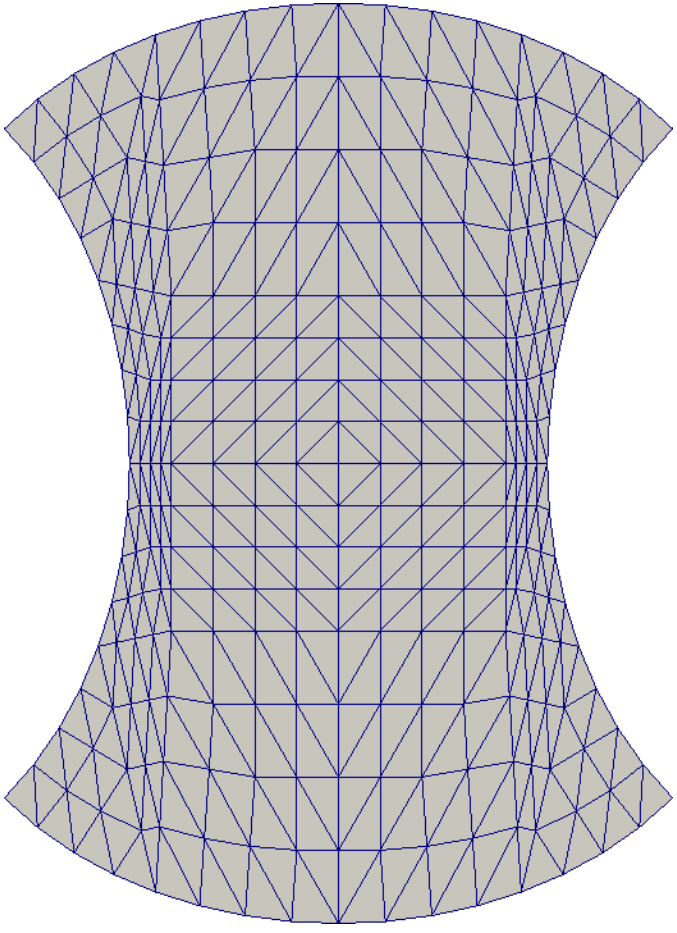


Figure: initial mesh

Task: Find a mapping

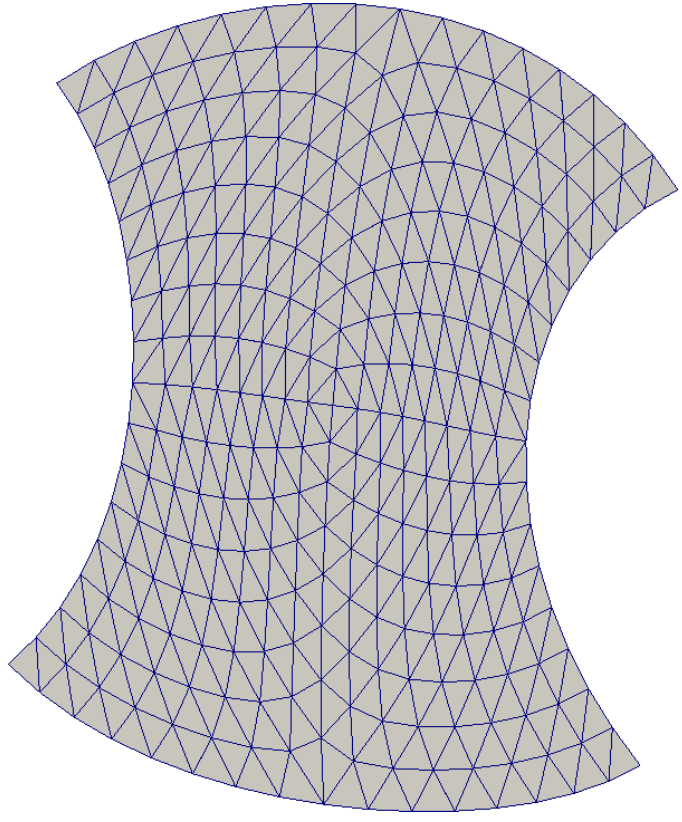


Figure: desired mesh

Link to solid mechanics

In solid mechanics

- engineers take an elastic material, for example a beam
- apply some load to it
- calculate the resulting stresses
- calculate the resulting strains
- calculate the deformed state
- and then calculate other quantities of interest

Sounds familiar!

Solid mechanics I

Well known:

- Relation between load/force and stresses
- Relation between strains and deformation

Unknown: The relation between stresses and strains!

Solid mechanics II

- Each material has an energy that models the material behaviour
- The internal energy of a material must not depend on the position in space
- All systems seek a state that minimizes the energy in the system
- There exist stable and unstable states that minimize the energy in the system

Unfortunately, the model cannot be linear!

Nonlinear model/Solid mechanics III

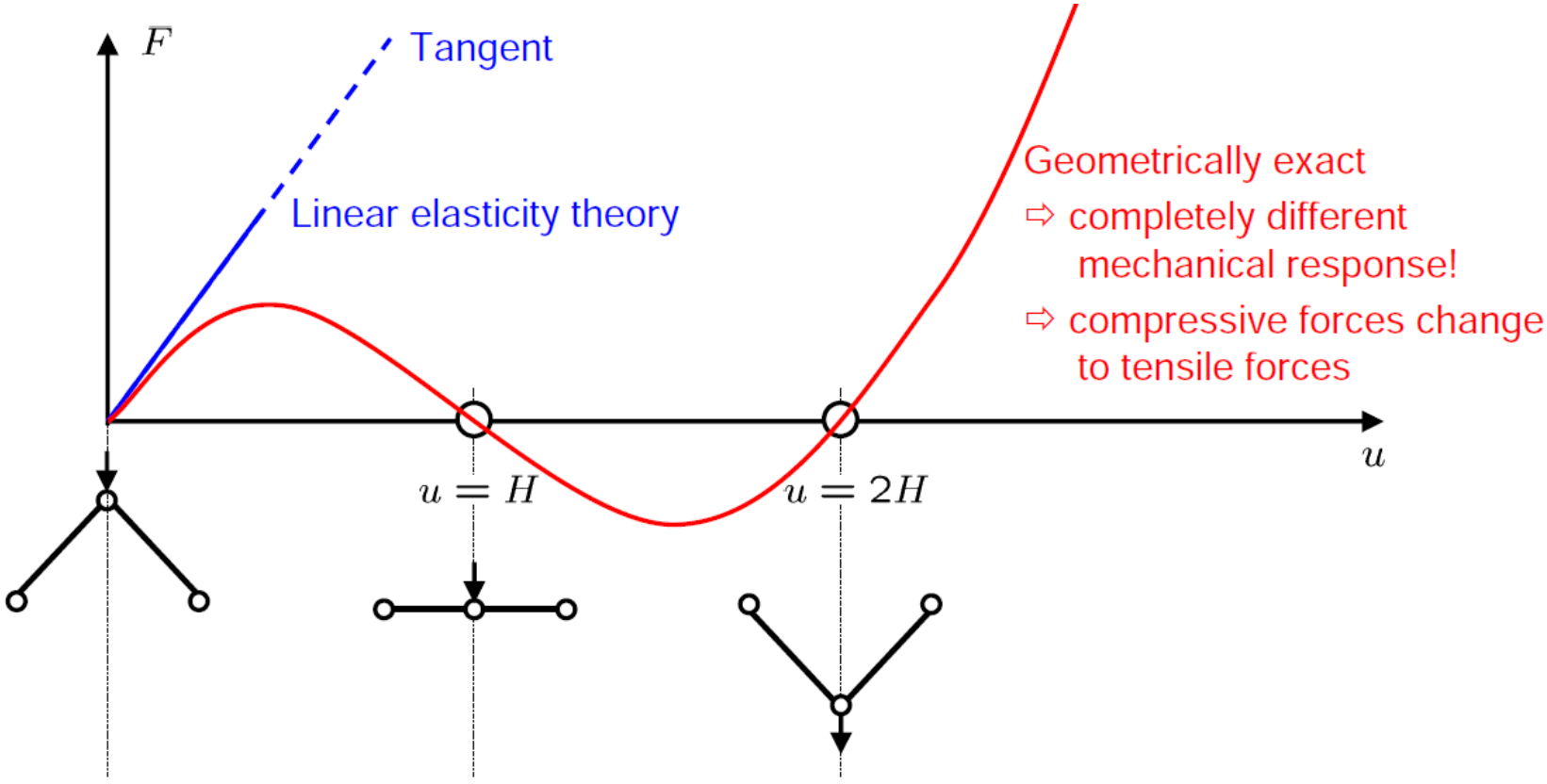


Figure: Taken from Skriptum by Prof. Mosler

Solid mechanics IV

The invariants of a Matrix A are defined as

- $\mathrm{Tr}(A)$
- $\frac{1}{2}((\mathrm{Tr}(A))^2 - \mathrm{Tr}(A^2))$
- $\det(A)$

Let $\varphi(X)$ be the mapping between the undeformed and deformed state. It can be proven that a model that is objective must depend on $\mathbf{C} := (\nabla\varphi)^T(\nabla\varphi)$.
Definition: A material model is called Neo-Hooke-Model if it only depends linear on $\mathrm{Tr}(\mathbf{C})$, not on $\frac{1}{2}((\mathrm{Tr}(\mathbf{C}))^2 - \mathrm{Tr}((\mathbf{C})^2))$ and in some way on $\det(\mathbf{C})$.

Solid mechanics IV

The invariants of a Matrix A are defined as

- $\mathrm{Tr}(A)$
- $\frac{1}{2}((\mathrm{Tr}(A))^2 - \mathrm{Tr}(A^2))$
- $\det(A)$

Let $\varphi(X)$ be the mapping between the undeformed and deformed state. It can be proven that a model that is objective must depend on $\mathbf{C} := (\nabla\varphi)^T(\nabla\varphi)$.

Definition: A material model is called Neo-Hooke-Model if it only depends linear on $\mathrm{Tr}(\mathbf{C})$, not on $\frac{1}{2}((\mathrm{Tr}(\mathbf{C}))^2 - \mathrm{Tr}((\mathbf{C})^2))$ and in some way on $\det(\mathbf{C})$.

Solid mechanics IV

The invariants of a Matrix A are defined as

- $\mathrm{Tr}(A)$
- $\frac{1}{2}((\mathrm{Tr}(A))^2 - \mathrm{Tr}(A^2))$
- $\det(A)$

Let $\varphi(X)$ be the mapping between the undeformed and deformed state. It can be proven that a model that is objective must depend on $\mathbf{C} := (\nabla\varphi)^T(\nabla\varphi)$.
Definition: A material model is called Neo-Hooke-Model if it only depends linear on $\mathrm{Tr}(\mathbf{C})$, not on $\frac{1}{2}((\mathrm{Tr}(\mathbf{C}))^2 - \mathrm{Tr}((\mathbf{C})^2))$ and in some way on $\det(\mathbf{C})$.

Solid mechanics V

- Stresses $\mathbf{P} = \frac{\partial \Psi(\nabla \varphi)}{\partial (\nabla \varphi)}$
- Initial configuration should be stress-free
- Growth condition: $\Psi(\nabla \varphi) \to \infty$ for $\det(\nabla \varphi) \to 0$
- Growth condition: $\Psi(\nabla \varphi) \to \infty$ for $\{||\nabla \varphi|| + ||\operatorname{cof}(\nabla \varphi)|| + \det(\nabla \varphi)\} \to \infty$

Note to stress-free

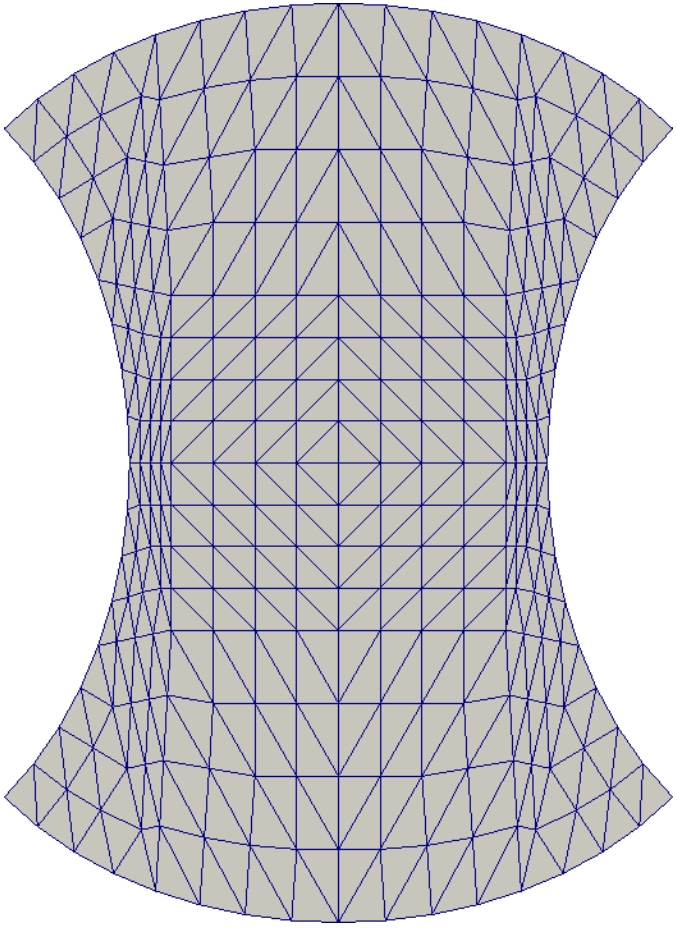


Figure: mesh from meshfile

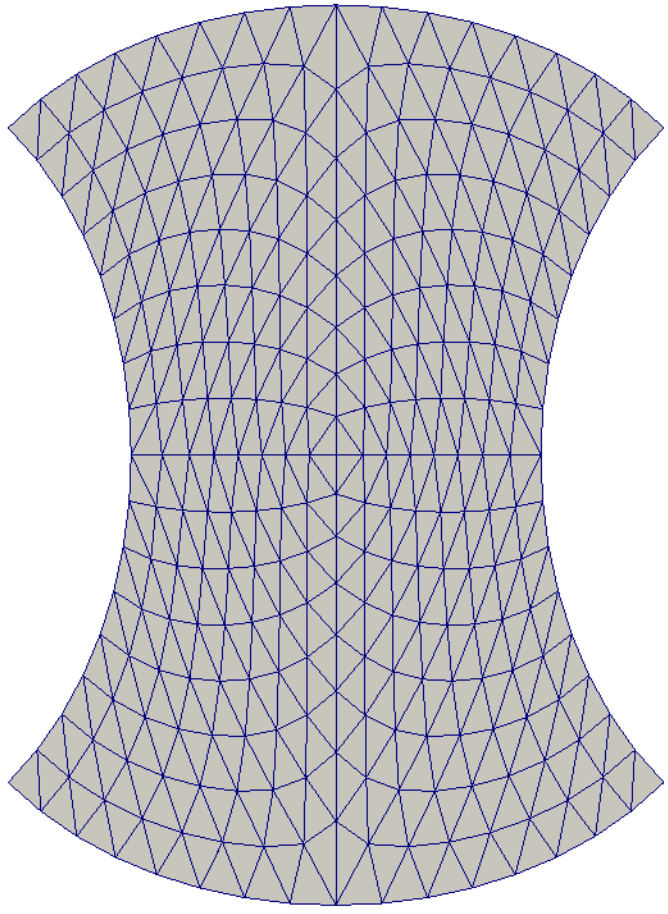


Figure: stress-free

Choice of energy

Standard Neo-Hookean-Type model:
$\Psi(\nabla\varphi) = \frac{1}{2}\lambda(\log(\sqrt{\det(\mathbf{C})}))^2 + \frac{1}{2}\mu(\mathrm{Tr}(\mathbf{C}) - d - 2\log(\sqrt{\det(\mathbf{C})}))$
with λ and μ material parameters (Lamé-Parameters) and d the spatial dimension

Semidiscrete level

Sum up all local contributions to the energy:

$$\Psi(\nabla\varphi) = \int \{\frac{1}{2}\lambda(\log(\sqrt{\det(\mathbf{C})}))^2$$

$$+ \frac{1}{2}\mu(\mathrm{Tr}(\mathbf{C}) - d - 2\log(\sqrt{\det(\mathbf{C})}))\}$$

This is the term we need to minimize $\rightarrow$ Nonlinear Optimization

General Descent Algorithm:
given: $F(x)$, x_0

Algorithm 6.1 General Descent Algorithm

find searchdirection s_k
find stepsize σ_k with $F(x_k + \sigma_k s_k) < F(x_k)$
$x_{k+1} = x_k + \sigma_k s_k$

Definition (searchdirection)

A direction s is called searchdirection to the point x and the function $F(x)$ if $\nabla F(x) \cdot s < 0$

Armijo-Scheme

Algorithm 6.2 Armijo-Scheme

given: x, searchdirection s
choose $\beta \in (0,1)$, $\gamma \in (0,1)$, $\sigma_0 > 0$, $k = 0$
while $f(x) - f(x + \sigma_k s) < -\gamma \sigma_k \nabla f(x)^T s$ **do**
  $\sigma_{k+1} = \beta \sigma_k$
  $k = k + 1$
end while

The Armijo-Scheme must always converge!
Attention: The dot-product must be independent of the mesh!

Choice of solver

- Gradient Method: $s_k = -\nabla F(x_k)$
- Nonlinear CG: $s_k = -\nabla F(x_k) + \beta_k s_{k-1}$
  Different choices of β_k possible (Fletcher-Reeves, Polak-Ribiere,...)

Testproblem

Testproblem:

- $\partial\Omega = \partial\Omega(t)$
- For $\Omega(t_{n+1})$: Use $\Omega(t_n)$ as initial mesh
- Deform until T using a timestep Δt

first results

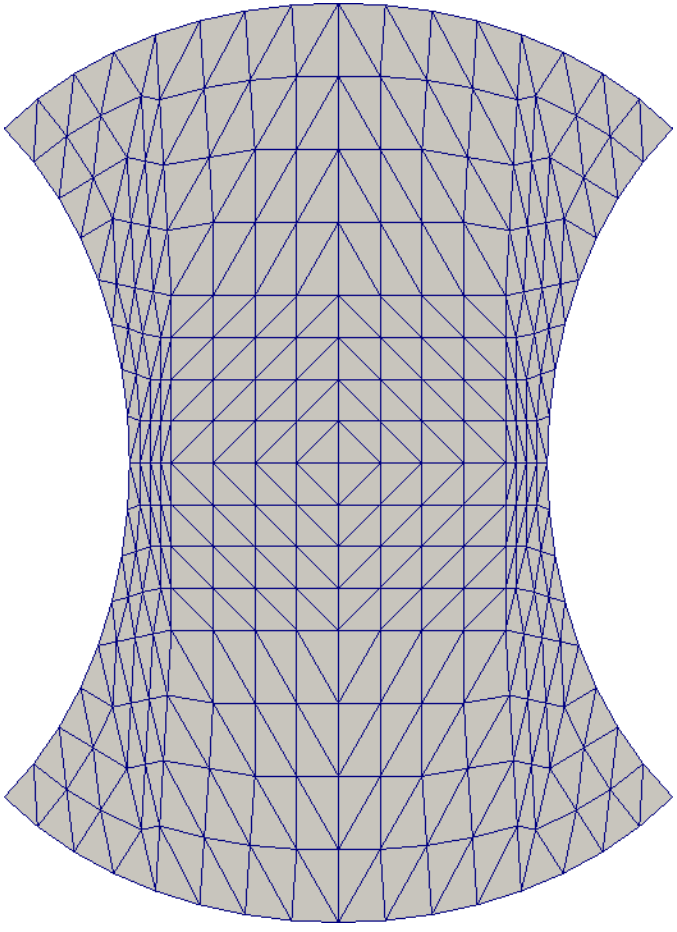


Figure: initial mesh

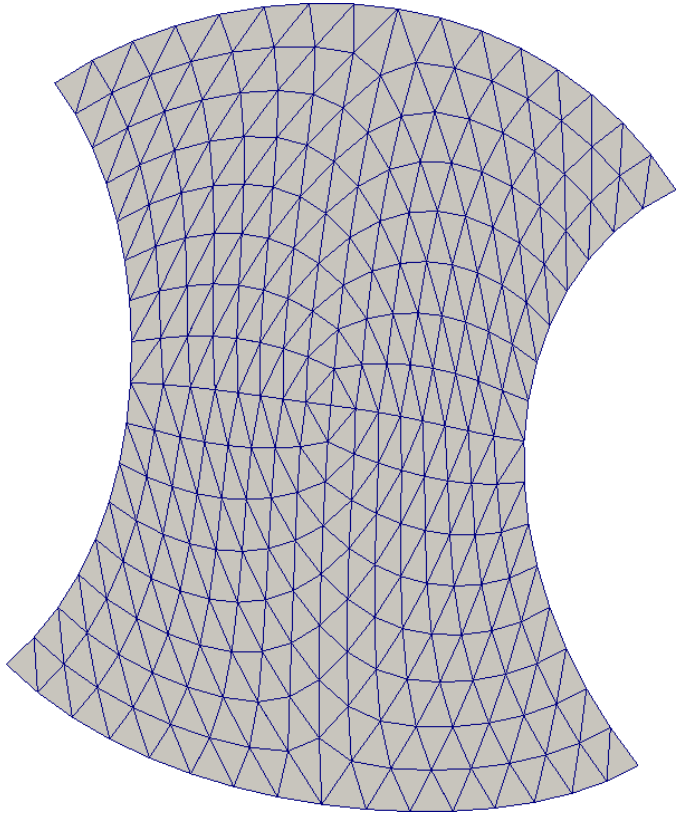
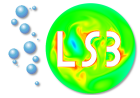


Figure: mesh after deformation

...first Problems



- NLCG needs more steps than Gradient Method
- Armijo-Scheme fails to compute a stepsize

Why? Implementation wrong? Solver not robust enough? Problem not well-posed?

Reason: Wrong gradient, but: wrong in other sense than you might think

...first Problems

- NLCG needs more steps than Gradient Method
- Armijo-Scheme fails to compute a stepsize

Why? Implementation wrong? Solver not robust enough? Problem not well-posed?
Reason: Wrong gradient, but: wrong in other sense than you might think

correct gradient

What do we do?

- Define an energy-functional analytical and discretize it
- Calculate the analytical derivative
- Discretize the analytical derivative
- Iterate until norm of derivative is small

However, we are not minimizing the analytical functional but the discretized version, so

$$\frac{\partial}{\partial u_i}\{\sum_K \sum_{x_i} \omega_i \frac{1}{2}\lambda(\log\sqrt{(\det \mathbf{C}(x_i))^2}\ldots\}$$

with u_i the vertices. Note: For $\mathbf{C}$ we also need to include all transformations...

Not practical!

correct gradient

What do we do?

- Define an energy-functional analytical and discretize it
- Calculate the analytical derivative
- Discretize the analytical derivative
- Iterate until norm of derivative is small

However, we are not minimizing the analytical functional but the discretized version, so

$$\frac{\partial}{\partial u_i}\{\sum_K \sum_{x_i} \omega_i \frac{1}{2}\lambda(\log\sqrt{(\det \mathbf{C}(x_i))^2}\ldots\}$$

with u_i the vertices. Note: For $\mathbf{C}$ we also need to include all transformations...

Not practical!

correct gradient

What do we do?

- Define an energy-functional analytical and discretize it
- Calculate the analytical derivative
- Discretize the analytical derivative
- Iterate until norm of derivative is small

However, we are not minimizing the analytical functional but the discretized version, so

$$\frac{\partial}{\partial u_i}\{\sum_K \sum_{x_i} \omega_i \frac{1}{2}\lambda(\log\sqrt{(\det \mathbf{C}(x_i))^2}\ldots\}$$

with u_i the vertices. Note: For $\mathbf{C}$ we also need to include all transformations...

Not practical!

gradient

Numerical gradient: Approximate the gradient with finite differences
Good idea, but

- the number of evaluations just to approximate the gradient is high
- this would be neccesary in each step of the Armijo-Scheme

Possible, but too expensive

gradient

Numerical gradient: Approximate the gradient with finite differences
Good idea, but

- the number of evaluations just to approximate the gradient is high
- this would be neccesary in each step of the Armijo-Scheme

Possible, but too expensive

Further investigation Ia

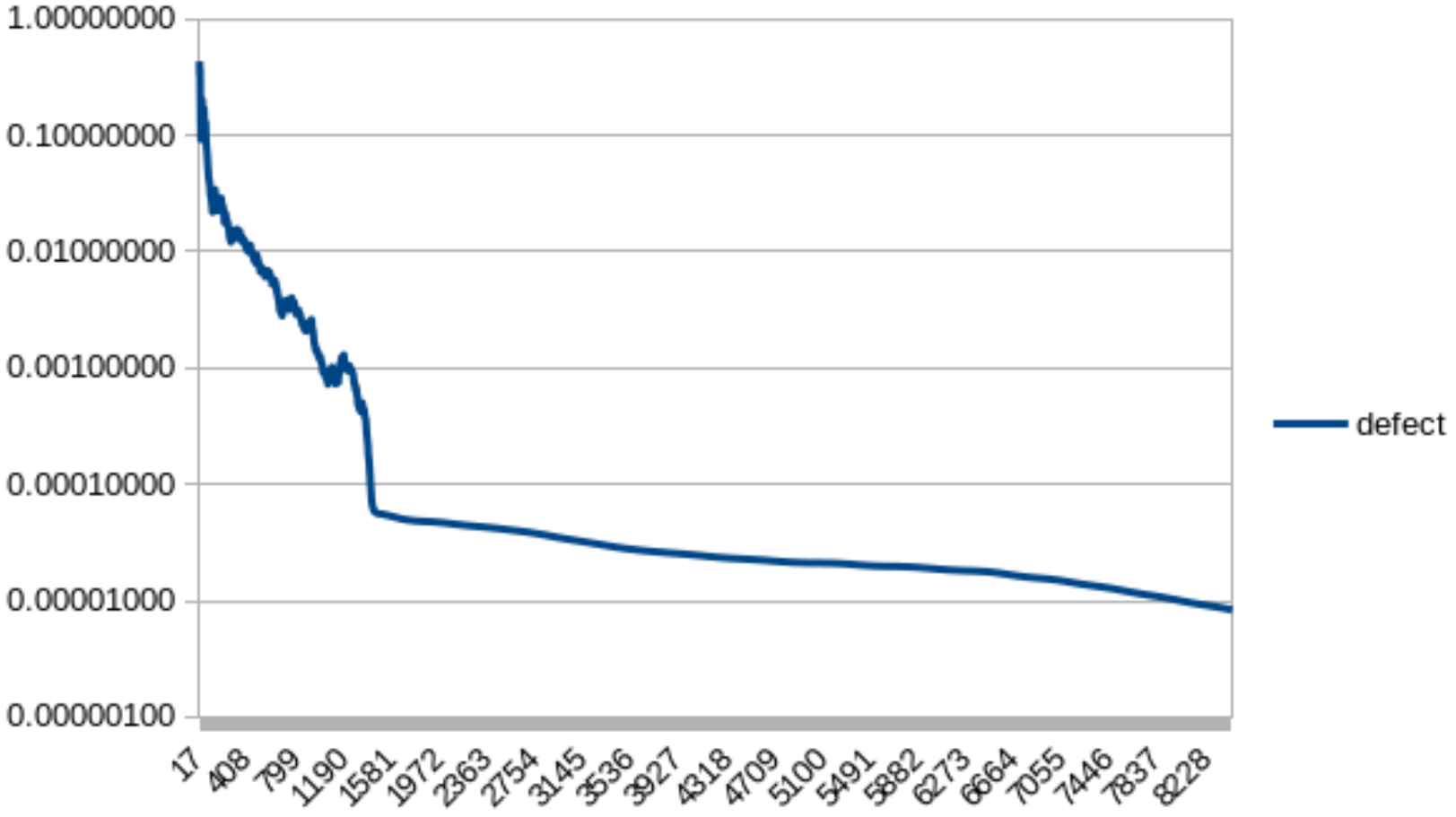


Figure: Defect for NLCG vs Iterations

Further investigation Ib

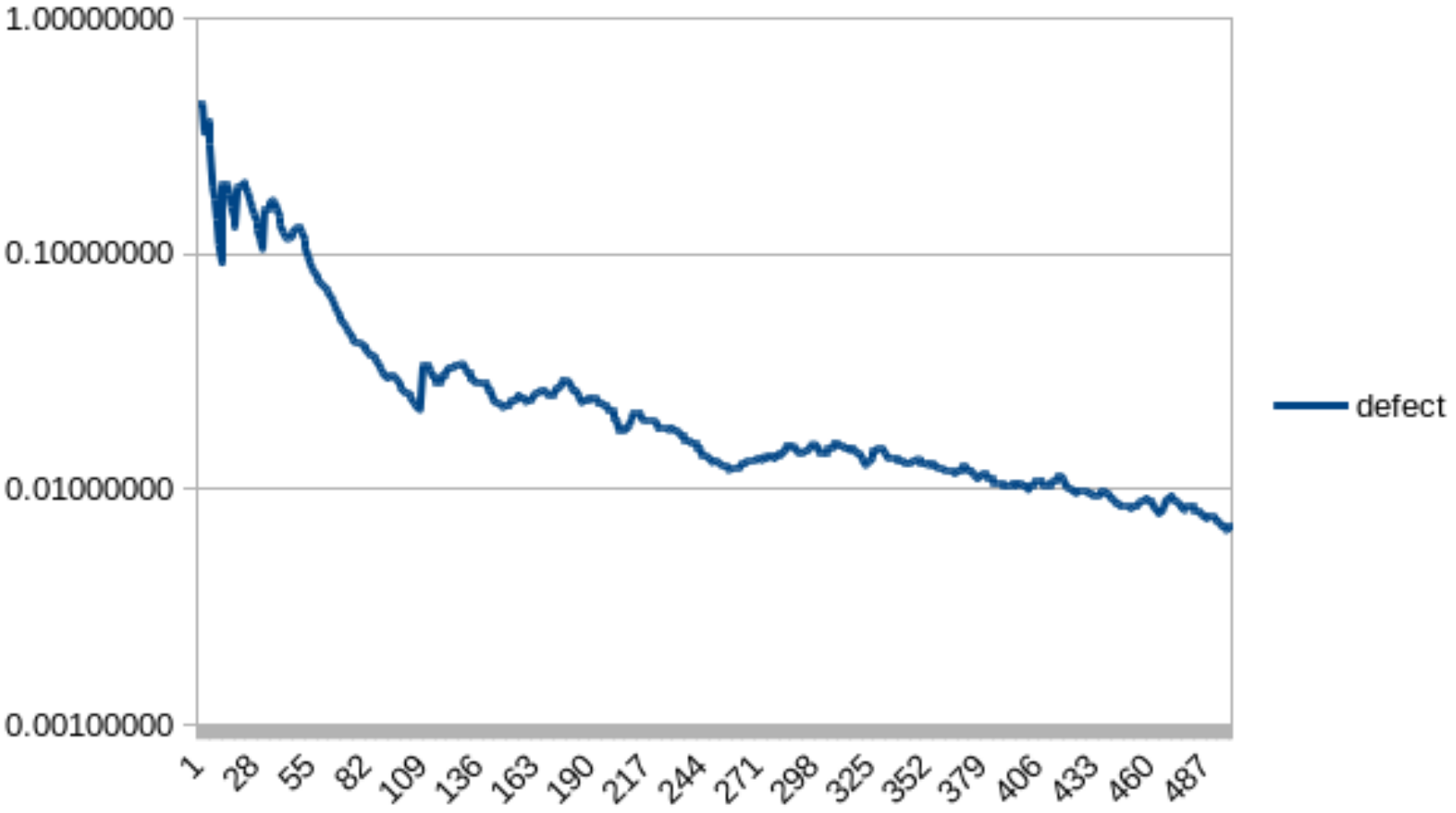


Figure: Defect for NLCG vs Iterations

Further investigation II

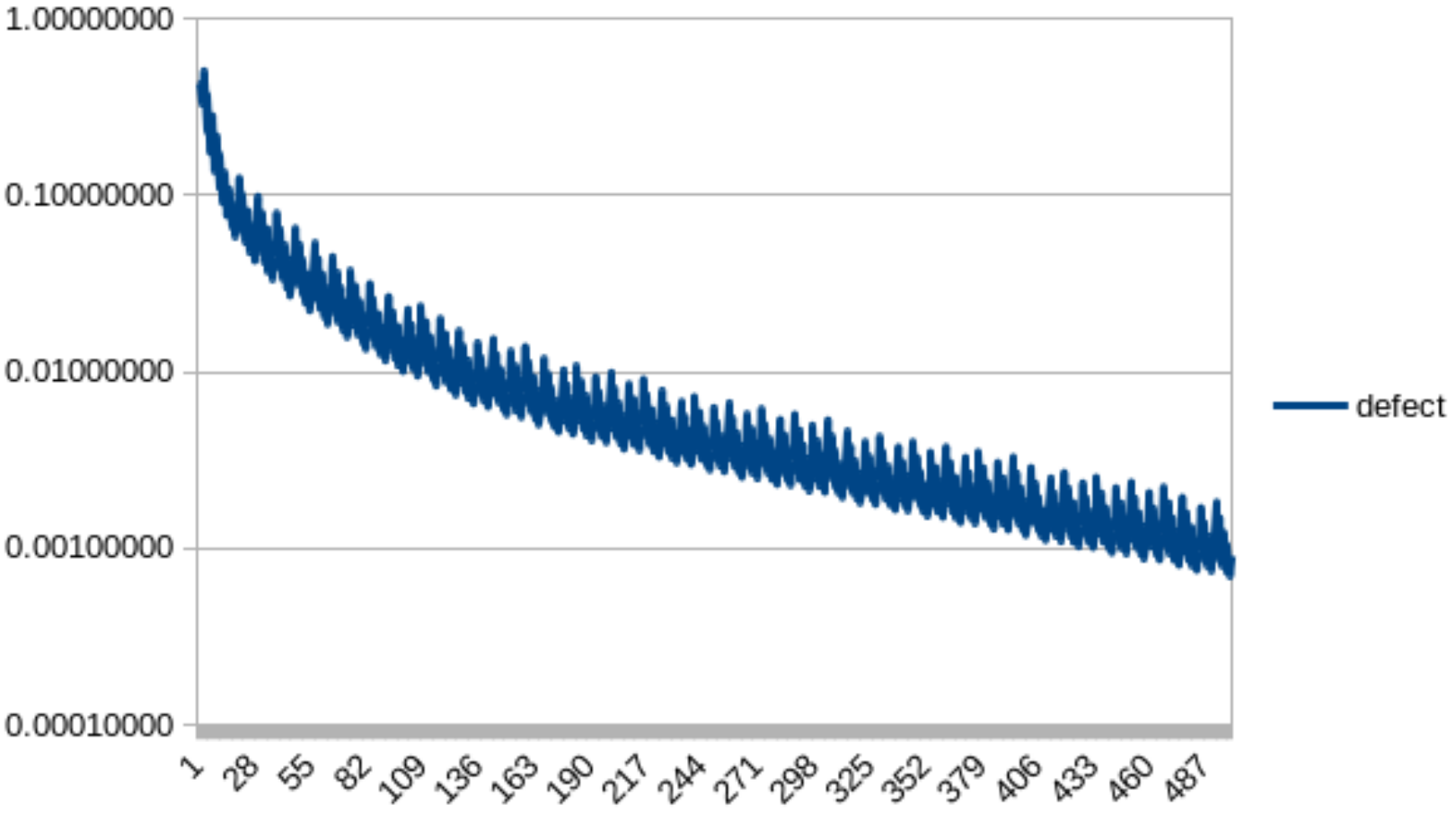


Figure: Defect for NLSD vs Iterations

Further investigation III

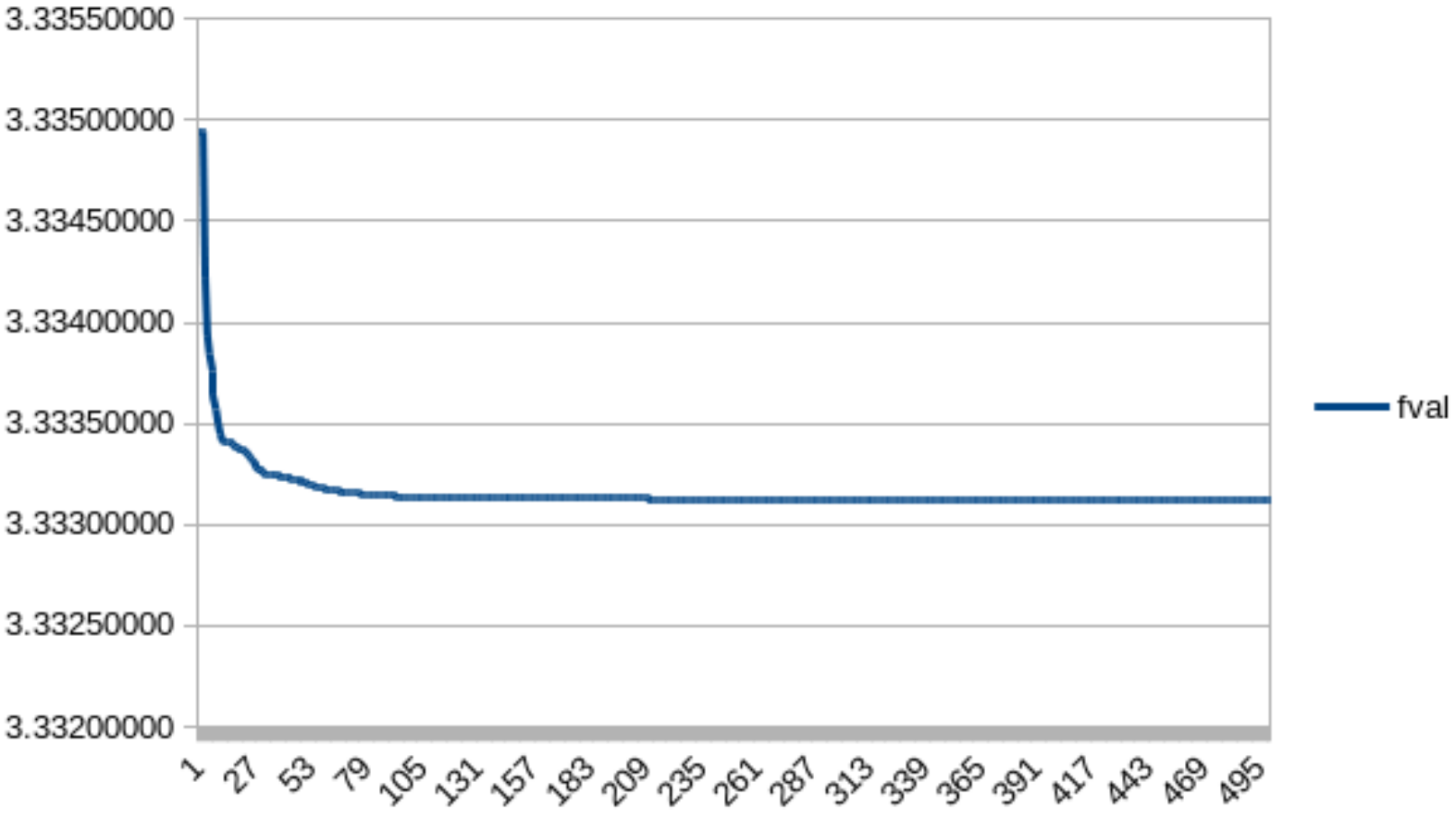


Figure: Functional value for NLCG vs Iterations

Further investigation IV

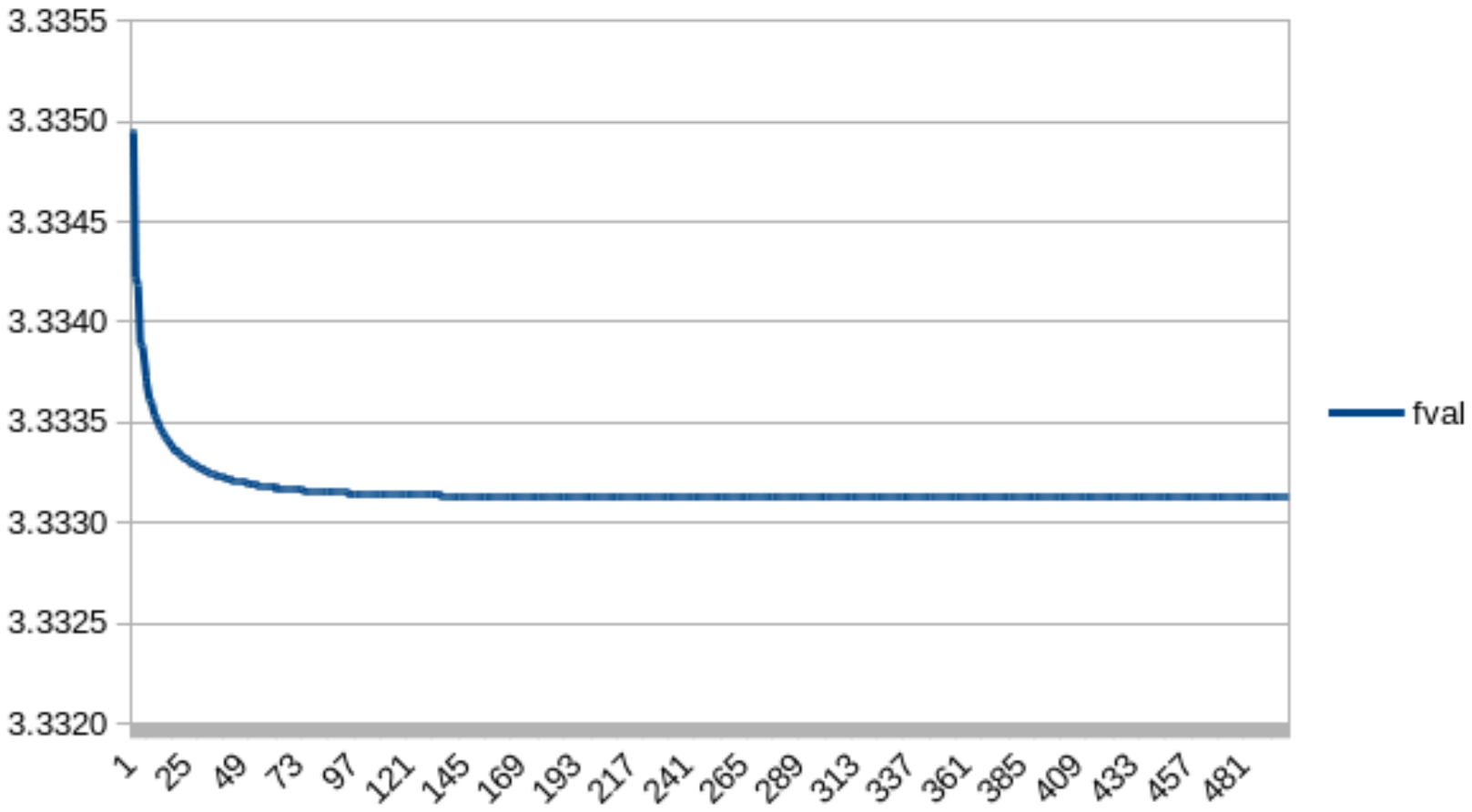


Figure: Functional value for NLSD vs Iterations

Mesh vs. Iterations

Idea: For $\Delta t = 10^{-3}$ calculate the mesh for $T = 0.28$. Limit the maximum number of iterations for each timestep drastical (earlier results needed over 1000 iterations per timestep)
Movies for NLCG and NLSD in seperate window

Modifications of the stopping-criteria

Ideas:

- Use a moderate relative stopping criterion, for example: Improve the residual by 10^{-2}.
  Problem: This is extremly hard to balance. On the one hand, you do not want too many iterations and run into the part where the mesh does not change any longer, on the other hand a too lose criterion will lead to irregular meshes after a few timesteps.
- Check for stagnation in the functional value.
  Problem: In my tests, the functional value stagnates relativly early: After 4 iterations, the value is changing only in the 5th digit, but visually the quality still improves when doing more iterations.

Modifications of the stopping-criteria

Ideas:

- Use a moderate relative stopping criterion, for example: Improve the residual by 10^{-2}.
  Problem: This is extremly hard to balance. On the one hand, you do not want too many iterations and run into the part where the mesh does not change any longer, on the other hand a too lose criterion will lead to irregular meshes after a few timesteps.
- Check for stagnation in the functional value.
  Problem: In my tests, the functional value stagnates relativly early: After 4 iterations, the value is changing only in the 5th digit, but visually the quality still improves when doing more iterations.

Modifications of the stopping-criteria

Ideas:

- Use a moderate relative stopping criterion, for example: Improve the residual by 10^{-2}.
  Problem: This is extremly hard to balance. On the one hand, you do not want too many iterations and run into the part where the mesh does not change any longer, on the other hand a too lose criterion will lead to irregular meshes after a few timesteps.
- Check for stagnation in the functional value.
  Problem: In my tests, the functional value stagnates relativly early: After 4 iterations, the value is changing only in the 5th digit, but visually the quality still improves when doing more iterations.

Modifications of the stopping-criteria

Ideas:

- Use a moderate relative stopping criterion, for example: Improve the residual by 10^{-2}.
  Problem: This is extremly hard to balance. On the one hand, you do not want too many iterations and run into the part where the mesh does not change any longer, on the other hand a too lose criterion will lead to irregular meshes after a few timesteps.
- Check for stagnation in the functional value.
  Problem: In my tests, the functional value stagnates relativly early: After 4 iterations, the value is changing only in the 5th digit, but visually the quality still improves when doing more iterations.

Modifications of the stopping-criteria

...

- Replace relative stopping criterion by an absolute.
  Problem: For large meshes this might be too hard to archive.
- Check for stagnation in the iterates
  Problem 1: For NLCG, the coordinates do not converge from one direction to the target but sometimes in a zick-zack. This would lead to too many iterations.
  Problem 2: If the coordinates converge not fast enough to the target the solver aborts too early.

Suggestions?

Modifications of the stopping-criteria

...

- Replace relative stopping criterion by an absolute.
  Problem: For large meshes this might be too hard to archive.
- Check for stagnation in the iterates
  Problem 1: For NLCG, the coordinates do not converge from one direction to the target but sometimes in a zick-zack. This would lead to too many iterations.
  Problem 2: If the coordinates converge not fast enough to the target the solver aborts too early.

Suggestions?

Modifications of the stopping-criteria

...

- Replace relative stopping criterion by an absolute.
  Problem: For large meshes this might be too hard to archive.
- Check for stagnation in the iterates
  Problem 1: For NLCG, the coordinates do not converge from one direction to the target but sometimes in a zick-zack. This would lead to too many iterations.
  Problem 2: If the coordinates converge not fast enough to the target the solver aborts too early.

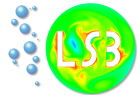
Suggestions?

Modifications of the stopping-criteria

...

- Replace relative stopping criterion by an absolute.
  Problem: For large meshes this might be too hard to archive.
- Check for stagnation in the iterates
  Problem 1: For NLCG, the coordinates do not converge from one direction to the target but sometimes in a zick-zack. This would lead to too many iterations.
  Problem 2: If the coordinates converge not fast enough to the target the solver aborts too early.

Suggestions?



Thank you for your attention!