

Mehrgitter und konjugierte Gradienten Verfahren

Juri Schröder, Alexander Schmidt

10.02.17

Inhaltsverzeichnis

- 1 Mehrgitter Verfahren
- 2 Verfahren der konjugierten Gradienten

Mehrgitter

```
function MG_ITERATION( $N$ ,  $p$ ,  $rhs$ )  
  SMOOTH( $N$ ,  $p$ ,  $rhs$ )  
  if  $N \leq 8$  then  
    SOLVE( $p$ ,  $rhs$ )  
  else  
     $res \leftarrow$  COMPUTE_RES( $N$ ,  $p$ ,  $rhs$ )  
     $res\_c \leftarrow$  RESTRICT( $N$ ,  $res$ )  
     $e\_c \leftarrow$  MG_ITERATION( $N/2$ ,  $e\_c$ ,  $res\_c$ )  
     $e \leftarrow$  INTERPOLATE( $N$ ,  $e\_c$ )  
     $p \leftarrow$  ADD( $p$ ,  $e$ )  
    SMOOTH( $N$ ,  $p$ ,  $rhs$ )
```

Mehrgitter Implementierung:

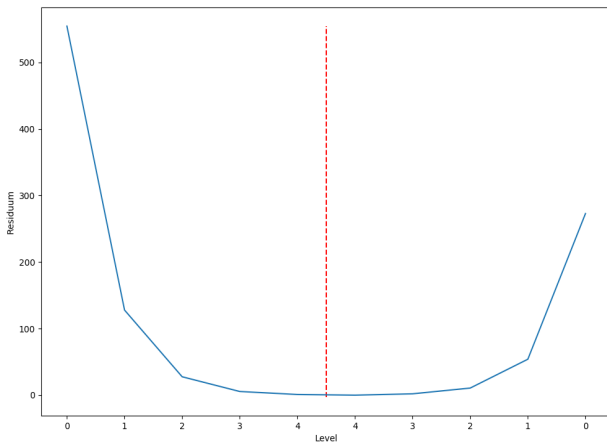
Generell:

- Glätten mit 2 Gauß-Seidel-Iterationen
- Lösen sobald $N \leq 8$ mit SOR

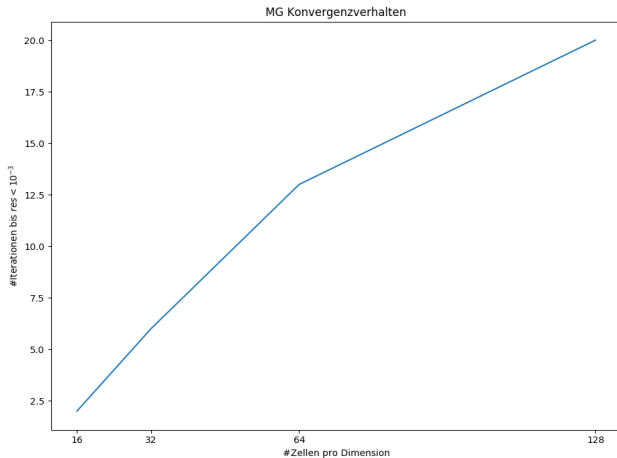
Parallelisierung:

- RedBlack-GS und RedBlackSOR
- Boundary-Austausch nach jedem Glätte- / Löser-Zyklus

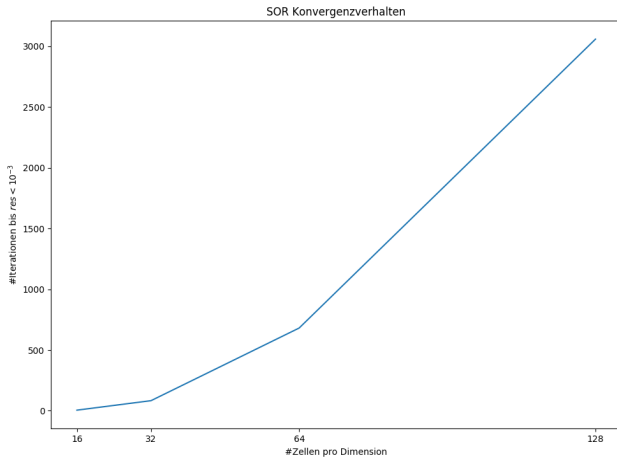
Mehrgitter Konvergenz



Mehrgitter Konvergenzanalyse

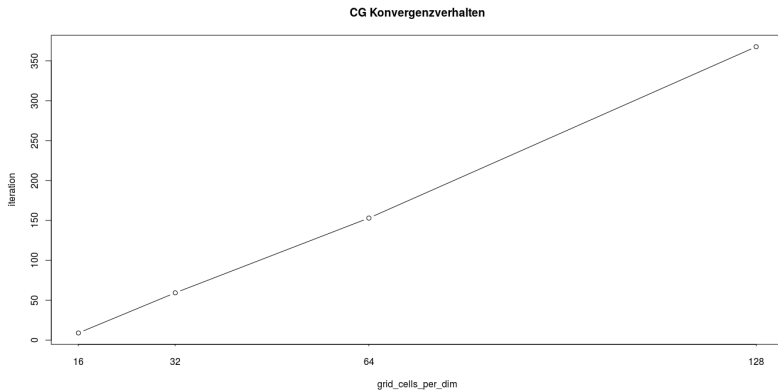


Vergleich mit SOR

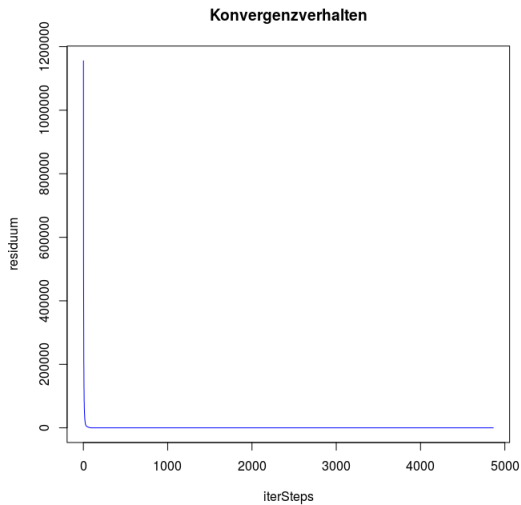


```
function CG::CYCLE( p, res, dir)
  for all InteriorIterator it do
    Ad.CELL(it)  $\leftarrow$  dir.DXX(it) + dir.DYY(it);
  res_old  $\leftarrow$  res.DOTPRODUCT(res)
  alpha  $\leftarrow$  res_old / dir.DOTPRODUCT(Ad)
  p  $\leftarrow$  p + alpha * dir;
  res  $\leftarrow$  res - alpha * Ad;
  res_new  $\leftarrow$  res.DOTPRODUCT( res );
  beta = res_old / res_new;
  dir  $\leftarrow$  dir + beta * res;
return res_new;
```

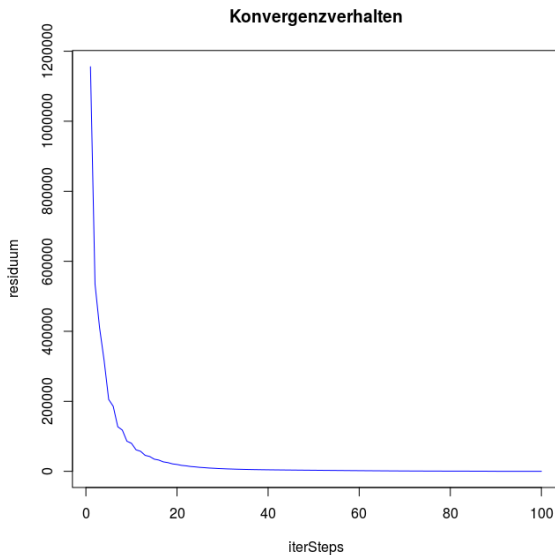

Konvergenzverhalten



Konvergenzverhalten



Konvergenzverhalten



Konvergenzverhalten

