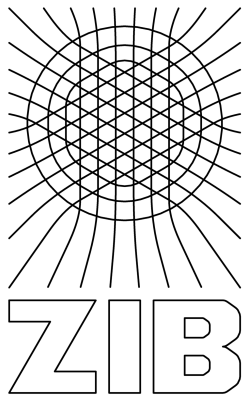
Solving Optimal Control Problems: Trajectory Compression & Implementation

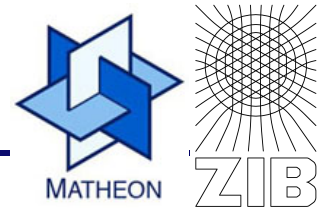
Sebastian Götschel, Martin Weiser



Zuse Institute Berlin

DUNE User Meeting Stuttgart, October 2010

Outline

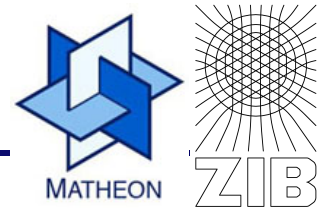


Trajectory Compression

Implementation

Examples

Outline



Trajectory Compression

Implementation

Examples

Optimal Control of Parabolic PDEs

$$\min J(y, u) \text{ subject to parabolic PDE } c(y, u) = 0$$

Assume

- $y = y(u)$ (locally) unique solution of state equation
- $c_y(y, u)$ admits a bounded inverse

Optimal Control of Parabolic PDEs

$$\boxed{\min J(y,u) \text{ subject to parabolic PDE } c(y,u) = 0}$$

Assume

- $y = y(u)$ (locally) unique solution of state equation
- $c_y(y,u)$ admits a bounded inverse

Then

- Equivalent reduced formulation: $\min j(u) = J(y(u),u)$
- reduced gradient $\nabla j(u) = J_u(y,u) - c_u(y,u)^\star c_y(y,u)^{-\star} J_y(y,u)$
- adjoint equation $c_y(y,u)^\star \lambda = J_y(y,u)$

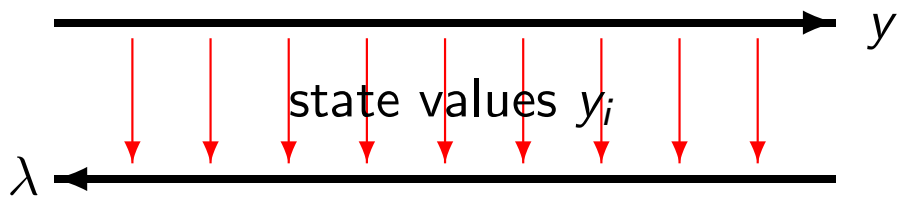
Optimal Control of Parabolic PDEs

Example

$$\min \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega\times(0,T))} + \frac{\alpha}{2}\|u\|^2_{L^2(\partial\Omega\times(0,T))}$$

$$\text{s.t.} \quad y_t - \Delta y = 0, \qquad \partial_\nu y + y = u, \quad y(\cdot,0) = 0$$

$$\Rightarrow -\lambda_t - \Delta\lambda = y - y_d, \quad \partial_\nu\lambda + \lambda = 0, \quad \lambda(\cdot,T) = 0$$



- adjoint equation is backward in time
- forward solution enters adjoint equation

storage of 4D data (mesh, state values) expensive: need data compression

A Lossy Compression Approach

Idea

- reduced gradient inexact due to discretization, iterative solution of linear systems

$$\delta j = \nabla j(u) + e \qquad \|e\| \leq \varepsilon \, \|\delta j\|$$

- allow small errors in the stored state values

A Lossy Compression Approach

Idea

- reduced gradient inexact due to discretization, iterative solution of linear systems

$$\delta j = \nabla j(u) + e \qquad \|e\| \leq \varepsilon \, \|\delta j\|$$

- allow small errors in the stored state values

Requirements

- control of error in the reduced gradient
- computationally inexpensive compression and reconstruction

A Lossy Compression Approach

Idea

- reduced gradient inexact due to discretization, iterative solution of linear systems

$$\delta j = \nabla j(u) + e \qquad \|e\| \leq \varepsilon \|\delta j\|$$

- allow small errors in the stored state values

Requirements

- control of error in the reduced gradient
- computationally inexpensive compression and reconstruction

For now: keep error of reconstructed states below prescribed tolerance

Stationary Compression

Setting

- nested family of triangulations $\mathcal{T}_0 \subset \cdots \subset \mathcal{T}_l$ of $\Omega \subset \mathbb{R}^d$
- at a fixed time step: solution of elliptic subproblem $y \in S_l$

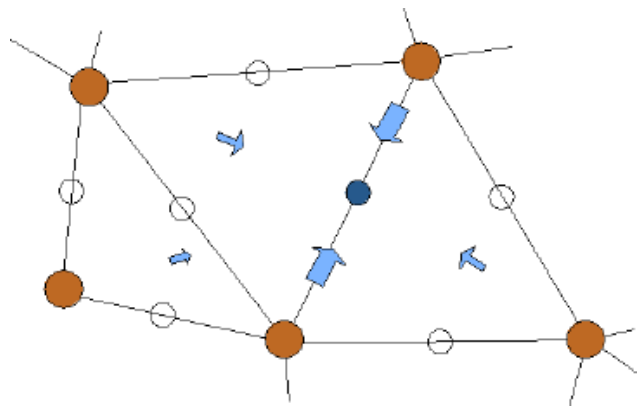
Stationary Compression

Setting

- nested family of triangulations $\mathcal{T}_0 \subset \cdots \subset \mathcal{T}_l$ of $\Omega \subset \mathbb{R}^d$
- at a fixed time step: solution of elliptic subproblem $y \in S_l$

Prediction in space

- multilevel prolongation in hierarchic meshes
- use reconstructed values for prediction



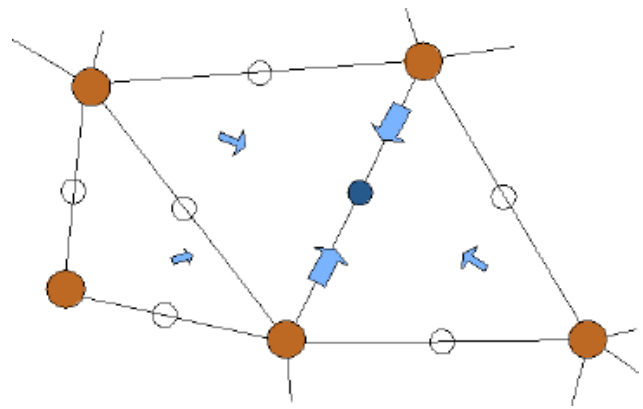
Stationary Compression

Setting

- nested family of triangulations $\mathcal{T}_0 \subset \cdots \subset \mathcal{T}_l$ of $\Omega \subset \mathbb{R}^d$
- at a fixed time step: solution of elliptic subproblem $y \in S_l$

Prediction in space

- multilevel prolongation in hierarchic meshes
- use reconstructed values for prediction



Quantization of prediction errors

- $I_j = [e_j^{\min}, e_j^{\max}]$ subdivided into $N_j = \frac{|I_j|}{2\delta}$ intervals
- Quantization error: $|y_k - \hat{y}_k| < \delta$

Stationary Compression

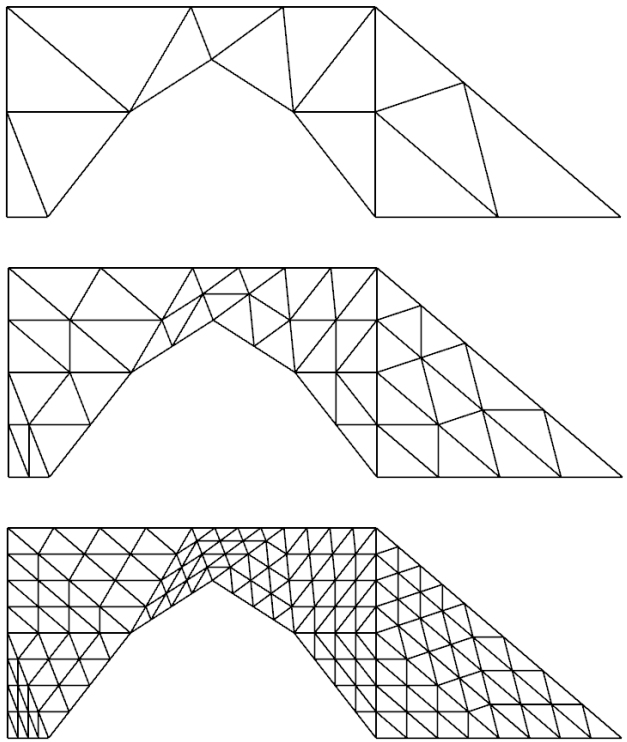
Algorithm

set $\hat{y}_{-1} = 0$

for $j = 0, \ldots, l$ do:

1. prolongation $\tilde{y}_j = P_j \hat{y}_{j-1}$
2. calculate prediction error $e_j = y_j - \tilde{y}_j$
3. quantize $q_j = Q(e_j)$
4. reconstruct $\hat{y}_j = \tilde{y}_j + Q^{\dagger}(q_j)$

write q to disk using entropy coding



Stationary Compression

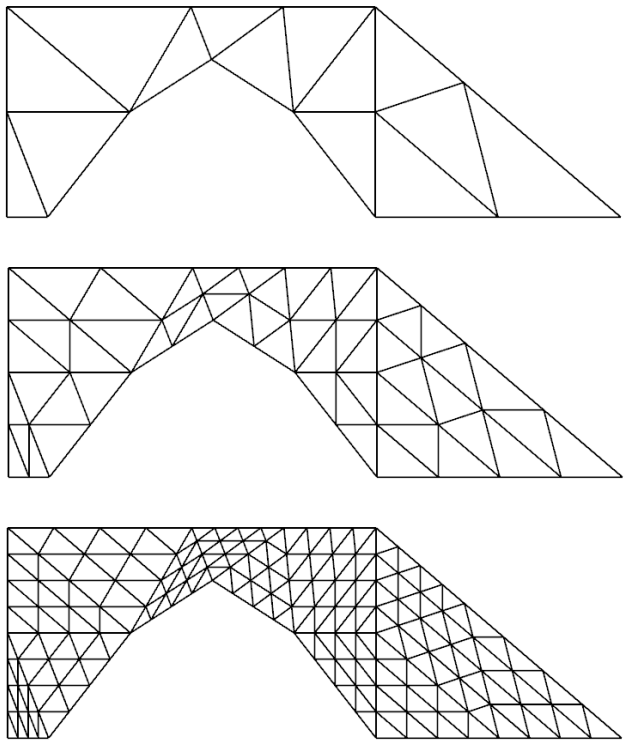
Algorithm

set $\hat{y}_{-1} = 0$

for $j = 0, \ldots, l$ do:

1. prolongation $\tilde{y}_j = P_j \hat{y}_{j-1}$
2. calculate prediction error $e_j = y_j - \tilde{y}_j$
3. quantize $q_j = Q(e_j)$
4. reconstruct $\hat{y}_j = \tilde{y}_j + Q^{\dagger}(q_j)$

write q to disk using entropy coding



Computational complexity

(less than) half a multigrid V-cycle

A Priori Estimates

Interpolation errors

Interpolation by polynomials of degree p for $y \in H^{p+1}$

$$\|y - I_j y\|_{m,h_j} \leq c h_j^{(p+1)-m} |y|_{p+1}$$

L^2-interpolation error between two grid levels for uniform refinement:

$$\|I_{j+1} y - I_j y\|_0 \leq c \frac{1}{2^{(p+1)j}} |y|_{p+1}$$

A Priori Estimates

Interpolation errors

Interpolation by polynomials of degree p for $y \in H^{p+1}$

$$\|y - I_j y\|_{m,h_j} \leq c h_j^{(p+1)-m} |y|_{p+1}$$

L^2-interpolation error between two grid levels for uniform refinement:

$$\|I_{j+1} y - I_j y\|_0 \leq c \frac{1}{2^{(p+1)j}} |y|_{p+1}$$

Hierarchichal basis

$S_l = V_0 + \cdots + V_l$ with $V_j = \text{span}\{\varphi_k : k \in \mathcal{N}_j \setminus \mathcal{N}_{j-1}\}$

$y = \sum_l \sum_k a_{jk} \psi_{jk}$ and $\|(a_{jk})_k\| \leq c \|I_j y - I_{j-1} y\| \leq c 2^{-2(j-1)} |y|_2$

A Priori Estimates

Assume linear interpolation, quantization with interval length δ

- number of quantized values: $\frac{c2^{-2(j-1)}}{\delta}$
- bits needed: $\mathrm{ld}(c/\delta) - 2(j-1)$

A Priori Estimates

Assume linear interpolation, quantization with interval length δ

- number of quantized values: $\frac{c2^{-2(j-1)}}{\delta}$
- bits needed: $\mathrm{ld}(c/\delta) - 2(j-1)$

Achieving discretization error accuracy: $\mathrm{ld}(c/\delta) - 2((l+1)-1) = 0$

- on level j: $2(l-j+1)$ bits/value

A Priori Estimates

Assume linear interpolation, quantization with interval length δ

- number of quantized values: $\frac{c2^{-2(j-1)}}{\delta}$
- bits needed: $\mathrm{ld}(c/\delta) - 2(j-1)$

Achieving discretization error accuracy: $\mathrm{ld}(c/\delta) - 2((l+1)-1) = 0$

- on level j: $2(l-j+1)$ bits/value

New nodes on level $j > 0$: $c\left(2^{jd} - 2^{(j-1)d}\right)$

- total bits: $\sum_{j=1}^{l} c\left(2^{jd} - 2^{(j-1)d}\right) \cdot 2(l-j+1) \approx c2^{ld}\frac{2^{d+1}}{2^d-1}$
- in 2D: ≈ 2.7 bits/value

Temporal Correlations

Prediction in time

Access to state values only backwards in time

- given quantized coefficients $q(t_{i-1})$, $q(t_i)$
- constant predictor, error $d(t_i) = q(t_{i-1}) - q(t_i)$

New nodes at time t_i due to grid refinement: predict as zero, or interpolation

Temporal Correlations

Prediction in time

Access to state values only backwards in time

- given quantized coefficients $q(t_{i-1})$, $q(t_i)$
- constant predictor, error $d(t_i) = q(t_{i-1}) - q(t_i)$

New nodes at time t_i due to grid refinement: predict as zero, or interpolation

Advantages

- very simple and inexpensive
- prediction of quantized values avoids error accumulation, lossless

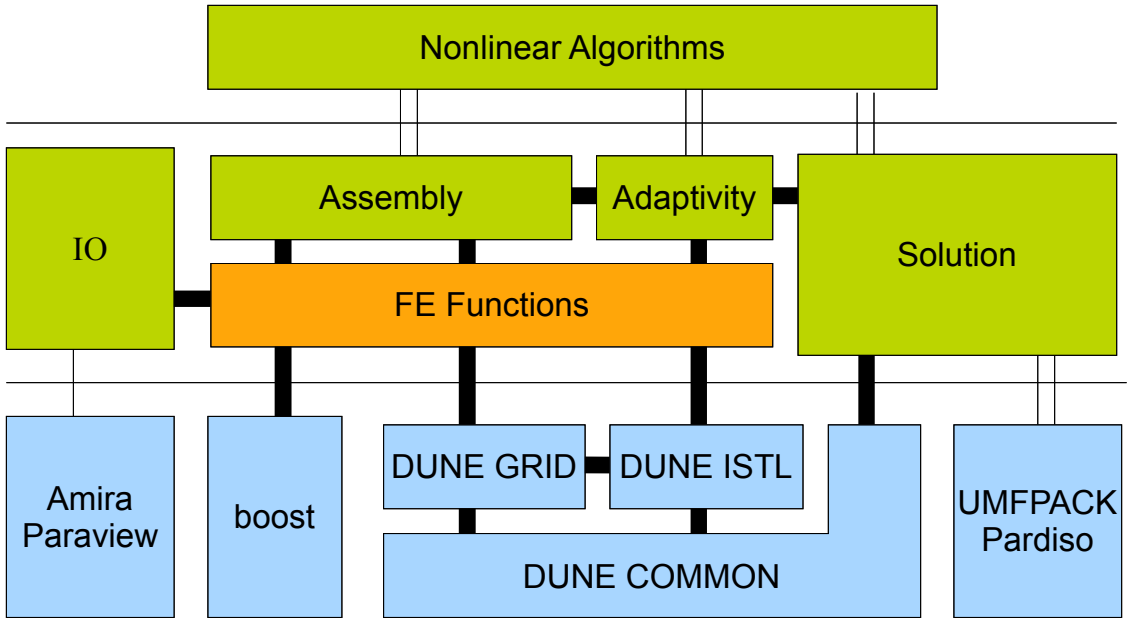
Outline

Trajectory Compression

Implementation

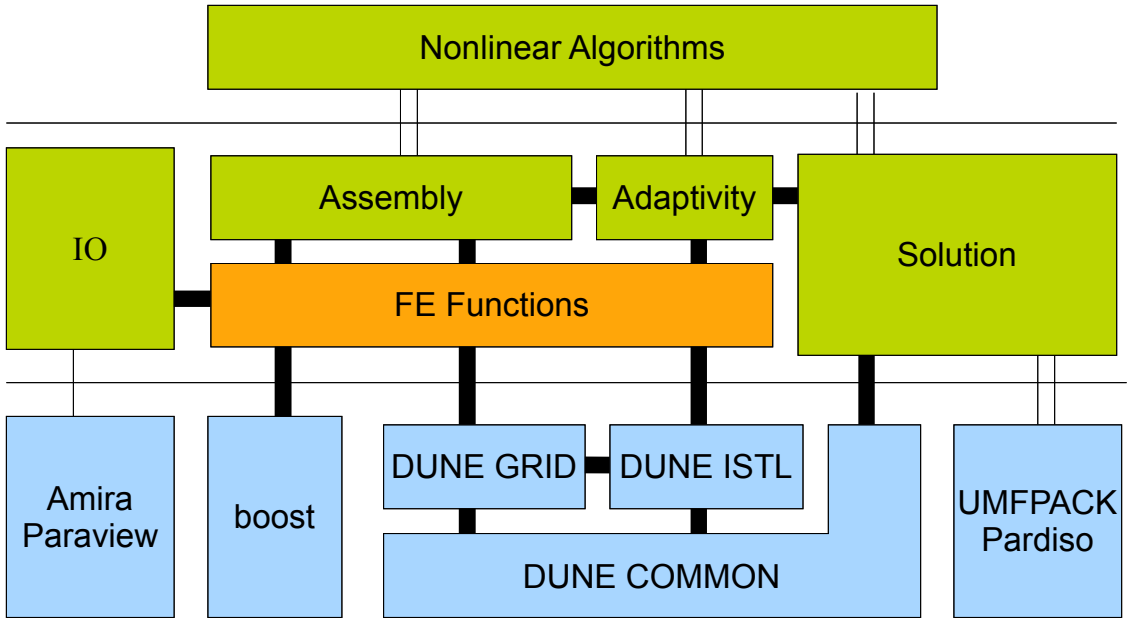
Examples

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



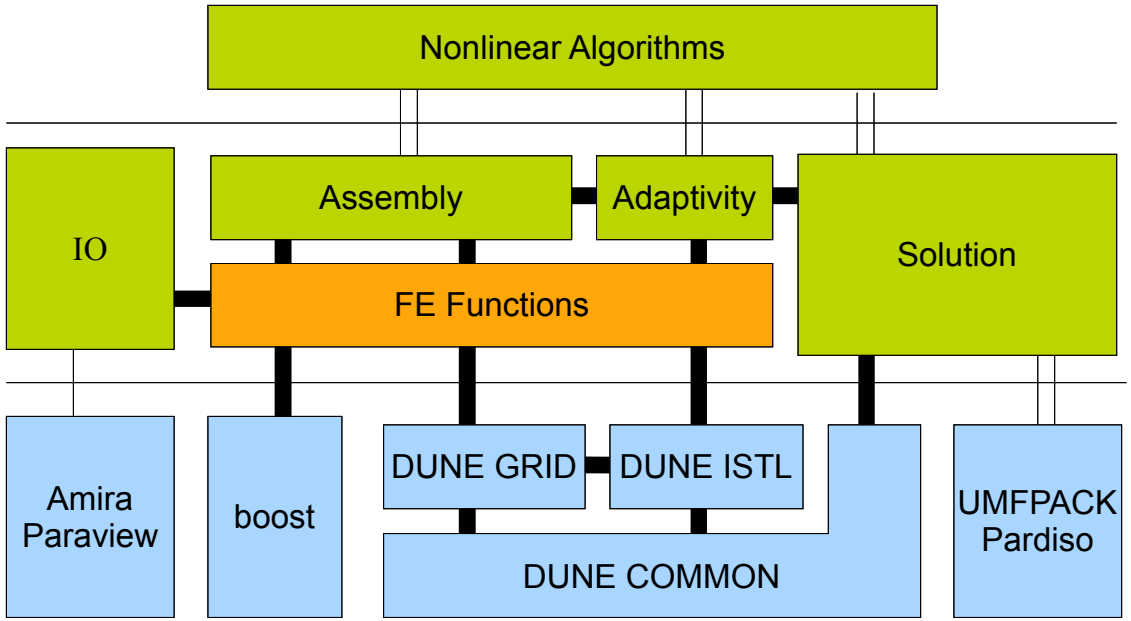
- finite element toolbox based on DUNE

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



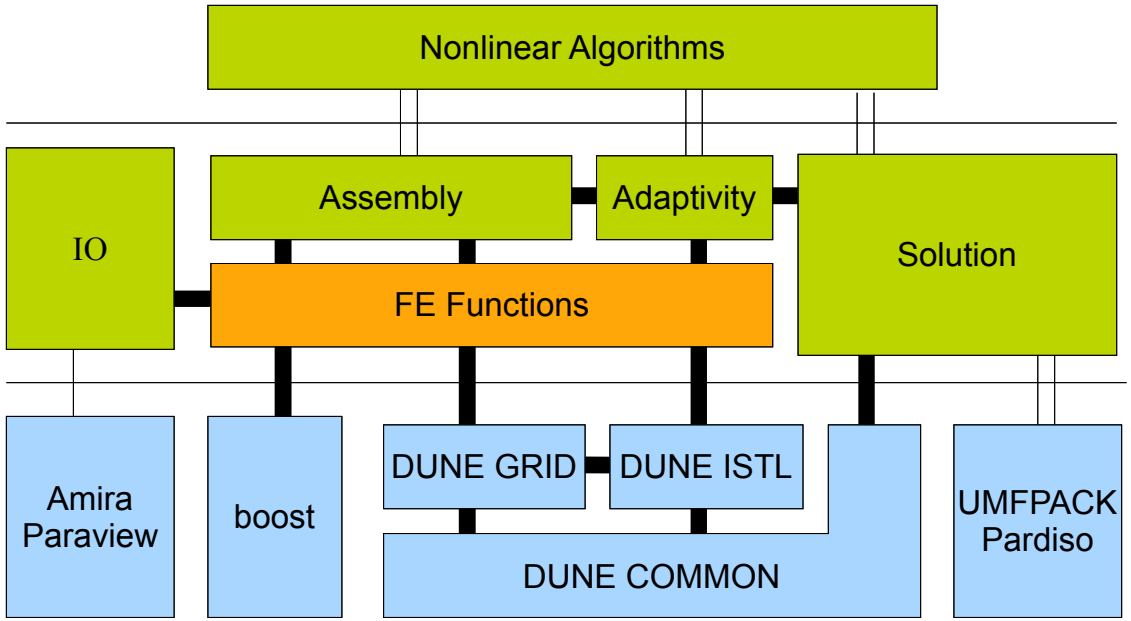
- spatial discretization
  - continuous & discontinuous, scalar & vectorial Lagrange elements
  - adaptive mesh refinement with hierarchical and embedded estimators

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



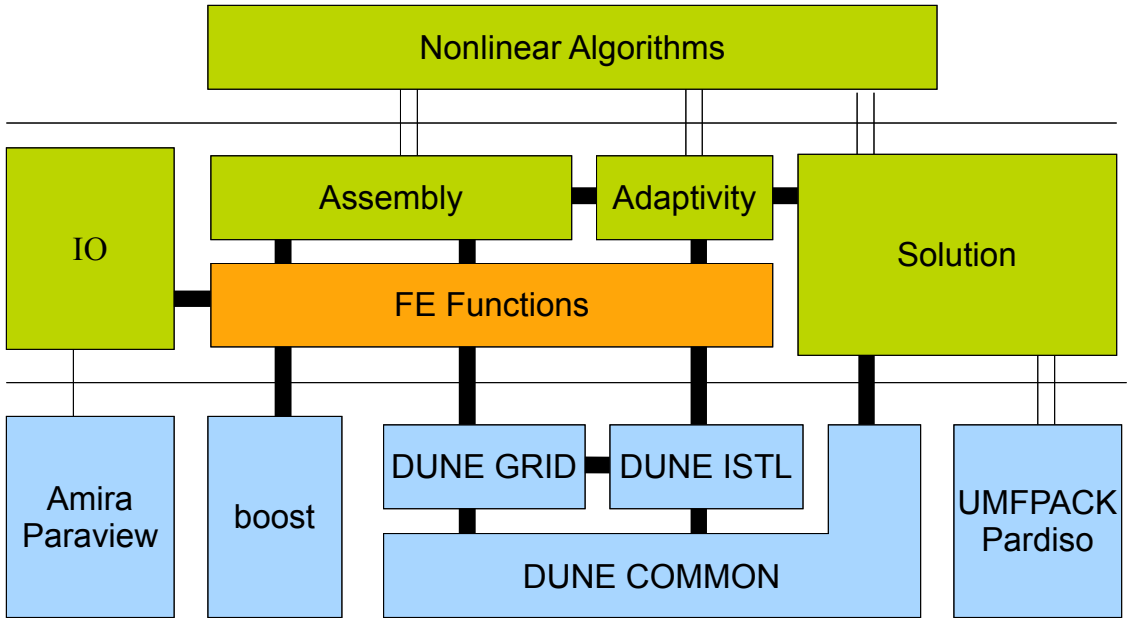
- time discretization
  - extrapolated linearly implicit Euler
  - adaptive time step selection

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



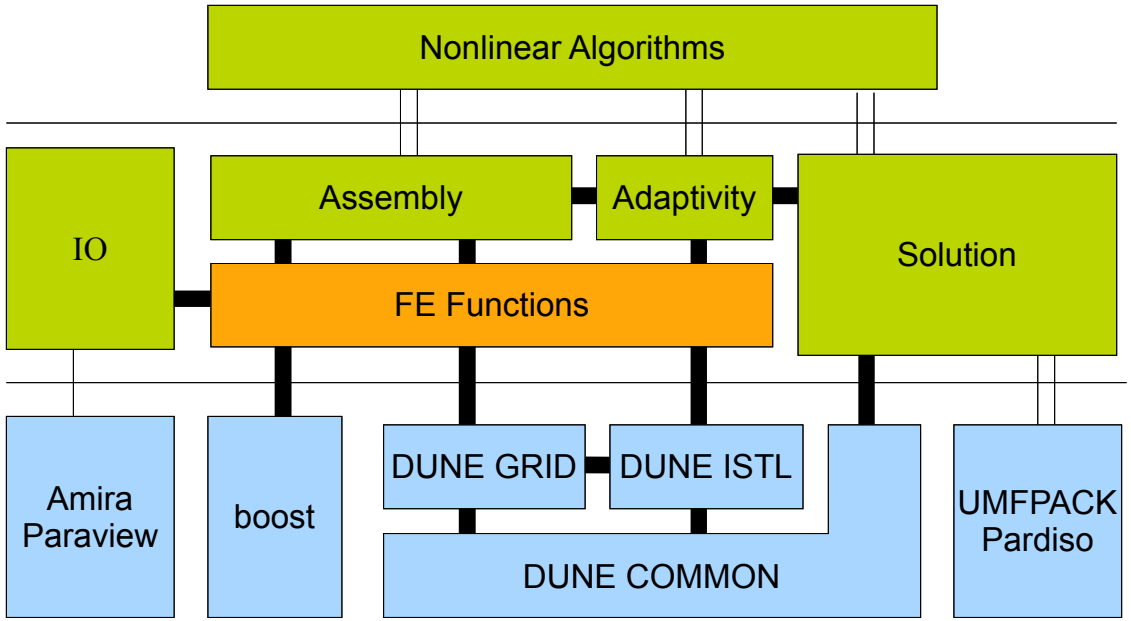
- interface to several linear systems solvers
  - direct solvers
  - iterative solution
  - preconditioning

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



- I/O
  - Triangle
  - Amira, VTK
  - compression of state values, mesh

KASKADE 7 [Weiser, Schiela (G., Erdmann, Roitzsch)]



- Nonlinear Algorithms
  - inexact Newton
  - interior point method

Problem Definition

Weak formulation of the (stationary) heat equation, Neumann BC

$$\int_\Omega \kappa \nabla y \cdot \nabla \varphi \mathrm{d}x = \int_\Omega f \varphi \mathrm{d}x + \int_\Gamma g \varphi \mathrm{d}s$$

Problem Definition

Weak formulation of the (stationary) heat equation, Neumann BC

$$\int_\Omega \kappa \nabla y \cdot \nabla \varphi \mathrm{d}x = \int_\Omega f\varphi \mathrm{d}x + \int_\Gamma g\varphi \mathrm{d}s$$

```
template<class Scalar, class VariableSet>
class HeatFunctional
{
   //...
   class DomainCache{ /*...*/ };
   class BoundaryCache{ /*...*/ };
};
```

Discretization in Space

```
typedef Dune::UGGrid<dim> Grid ;
std::auto_ptr<Grid> grid( new Grid() );
createGrid3d(grid, refinements);
GridManager<Grid> gridManager(grid);

typedef Grid::LeafGridView GridView ;
typedef ContinuousLagrangeMapper<double,GridView> Mapper ;
typedef FEFunctionSpace<Mapper> H1Space ;
```

Discretization in Space

```
typedef Dune::UGGrid<dim> Grid ;
std::auto_ptr<Grid> grid( new Grid() );
createGrid3d(grid, refinements);
GridManager<Grid> gridManager(grid);

typedef Grid::LeafGridView GridView ;
typedef ContinuousLagrangeMapper<double,GridView> Mapper ;
typedef FEFunctionSpace<Mapper> H1Space ;


GridView leafView = gridManager.grid().leafView() ;
H1Space temperatureSpace( gridManager, leafView, order) ;
```

Discretization in Space

```
typedef Dune::UGGrid<dim> Grid ;
std::auto_ptr<Grid> grid( new Grid() );
createGrid3d(grid, refinements);
GridManager<Grid> gridManager(grid);

typedef Grid::LeafGridView GridView ;
typedef ContinuousLagrangeMapper<double,GridView> Mapper ;
typedef FEFunctionSpace<Mapper> H1Space ;


GridView leafView = gridManager.grid().leafView() ;
H1Space temperatureSpace( gridManager, leafView, order) ;


typedef boost::fusion::vector<H1Space const*> Spaces ;
Spaces spaces(&temperatureSpace);
```

Discretization in Space

```
typedef Dune::UGGrid<dim> Grid ;
std::auto_ptr<Grid> grid( new Grid() );
createGrid3d(grid, refinements);
GridManager<Grid> gridManager(grid);

typedef Grid::LeafGridView GridView ;
typedef ContinuousLagrangeMapper<double,GridView> Mapper ;
typedef FEFunctionSpace<Mapper> H1Space ;


GridView leafView = gridManager.grid().leafView() ;
H1Space temperatureSpace( gridManager, leafView, order) ;


typedef boost::fusion::vector<H1Space const*, L2Space const*> Spaces ;
Spaces spaces(&temperatureSpace, &l2Space);
```

Discretization in Space

```
typedef boost::fusion::vector<VariableDescription<0,1,0> >
          VariableDescriptions;
std::string varNames[1] =  "T" ;

typedef VariableSetDescription<Spaces,VariableDescriptions>
          VariableSet;
VariableSet variableSet(spaces,varNames);
VariableSet::Representation y ;

typedef HeatEquation<double,VariableSet> Equation;
Equation eq;
```

Discretization in Space

```
typedef boost::fusion::vector<VariableDescription<0,1,0>,
          VariableDescription<1,3,1> > VariableDescriptions;
std::string varNames[1] =  "T" ;

typedef VariableSetDescription<Spaces,VariableDescriptions>
          VariableSet;
VariableSet variableSet(spaces,varNames);
VariableSet::Representation y ;

typedef HeatEquation<double,VariableSet> Equation;
Equation eq;
```

Discretization in Space

```
typedef boost::fusion::vector<VariableDescription<0,1,0> >
          VariableDescriptions;
std::string varNames[1] =  "T" ;

typedef VariableSetDescription<Spaces,VariableDescriptions>
          VariableSet;
VariableSet variableSet(spaces,varNames);
VariableSet::Representation y ;

typedef HeatEquation<double,VariableSet> Equation;
Equation eq;


// assemble and solve linear system
// or integrate(...) for time-dependent problems
```

Discretization in Time

based on linearly implicit Euler step for

$$B\dot{y} = F(y), \quad y(0) = y_0$$

```
Limex<Equation> limex(gridManager,eq,variableSet);
for (steps=0; !done && steps<maxSteps; ++steps) {
   do {
      dy = limex.step(y,dt,extrapolOrder,tolX);
      errors = limex.estimateError(/*...*/);
      //...
   } while( error > tolT );
   y += dy;
   writeStateToDisk(y,filename);
}
```

Lossy Storage

```
UniformQuantizationPolicy<VariableSet> quantizationPolicy ;
LossyStorage<UniformQuantizationPolicy<VariableSet> >
   lossyStorage( /*...*/, qTol, quantizationPolicy ) ;
```

Lossy Storage

```
UniformQuantizationPolicy<VariableSet> quantizationPolicy ;
LossyStorage<UniformQuantizationPolicy<VariableSet> >
   lossyStorage( /*...*/, qTol, quantizationPolicy ) ;


for (steps=0; !done && steps<maxSteps; ++steps) {
   do {
      dy = limex.step(y,dt,extrapolOrder,tolX);
      errors = limex.estimateError(/*...*/);
      //...
   } while( error > tolT );
   y += dy ;
   lossyStorage.encode(y,filename);
}
```

Hierarchic Error Estimator

- Given ansatz space V_l, discrete solution $A_{ll}x_l = -b_l$
- Extend ansatz space by another ansatz space V_h

$$\begin{pmatrix} A_{ll} & A_{lh} \\ A_{hl} & A_{hh} \end{pmatrix} \begin{pmatrix} x_l \\ x_h \end{pmatrix} = - \begin{pmatrix} b_l \\ b_h \end{pmatrix}$$

- solve reduced extension system $A_{hh}x_h = -(b_h + A_{hl}x_l)$

Hierarchic Error Estimator

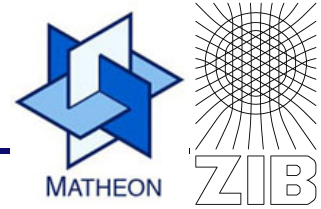
- Given ansatz space V_l, discrete solution $A_{ll}x_l = -b_l$
- Extend ansatz space by another ansatz space V_h

$$\begin{pmatrix} A_{ll} & A_{lh} \\ A_{hl} & A_{hh} \end{pmatrix} \begin{pmatrix} x_l \\ x_h \end{pmatrix} = - \begin{pmatrix} b_l \\ b_h \end{pmatrix}$$

- solve reduced extension system $A_{hh}x_h = -(b_h + A_{hl}x_l)$

```
typedef FEFunctionSpace<ContinuousHierarchicExtensionMapper</*...*/> >
            ExSpace;
//...
typedef HierarchicErrorEstimator<Linearization,ExVariableSet>
            ErrorEstimator;
typedef VariationalFunctionalAssembler<ErrorEstimator> EstGop;
EstGop estGop(gridManager.signals,exSpaces);
//...
estGop.assemble(ErrorEstimator(/*...*/));
Factorization<double> *matrix = new UMFFactorization<double>(/*...*/);
matrix->solve(estRhs,estSol);
```

Outline



Trajectory Compression

Implementation

Examples

Boundary Control

Boundary control of linear heat equation

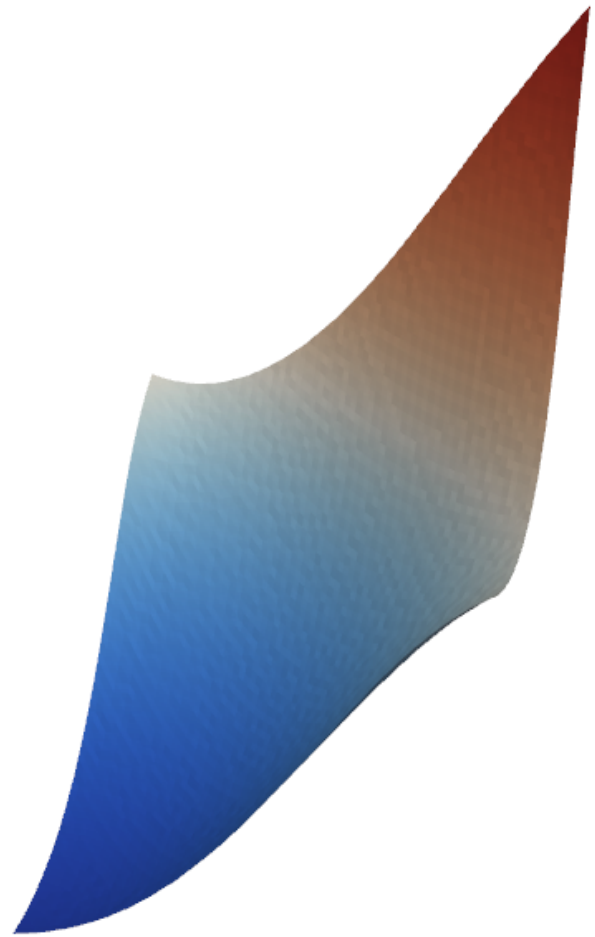
$$\min \frac{1}{2}\|y - y_d\|^2_{L^2(\Omega\times(0,T))} + \frac{\alpha}{2}\|u\|^2_{L^2(\partial\Omega\times(0,T))}$$

subject to

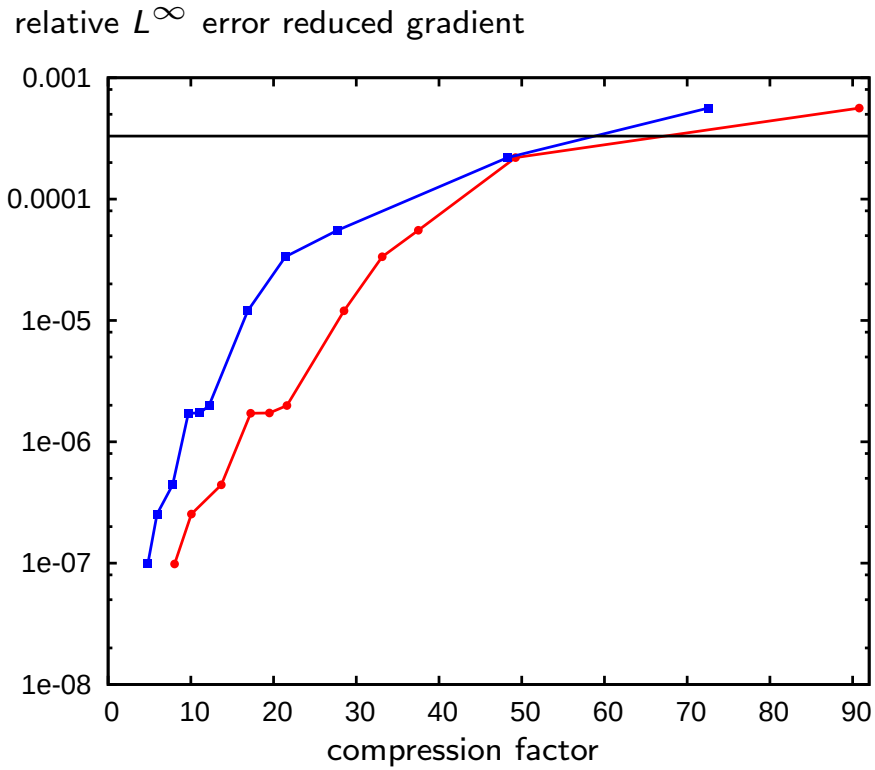
$$\begin{aligned} y_t - \Delta y &= f && \text{in } \Omega \times (0,T) \\ \partial_\nu y + y &= u && \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= 0 && \text{in } \Omega \end{aligned}$$

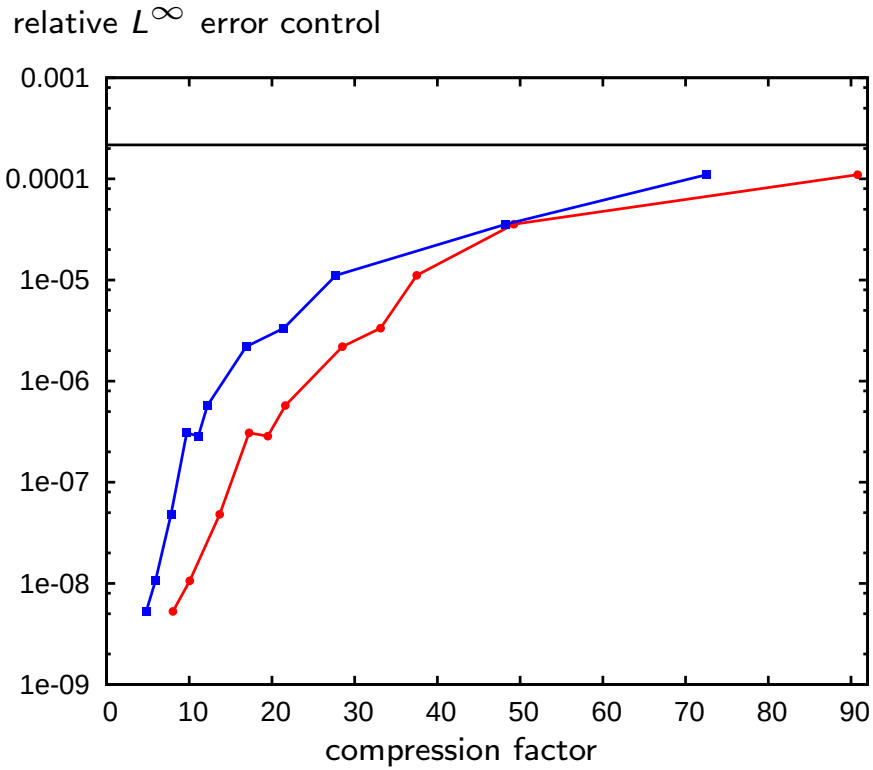
with data

$$\begin{aligned} \Omega = [0,1]\times[0,1],\ T &= 1, \\ y_d(x,t) &= t((x_0-1)^2 + (x_1-1)^2), \\ f(x,t) &= (x_0-1)^2 + (x_1-1)^2 - 4t. \end{aligned}$$



Boundary Control





without temporal prediction

with temporal prediction

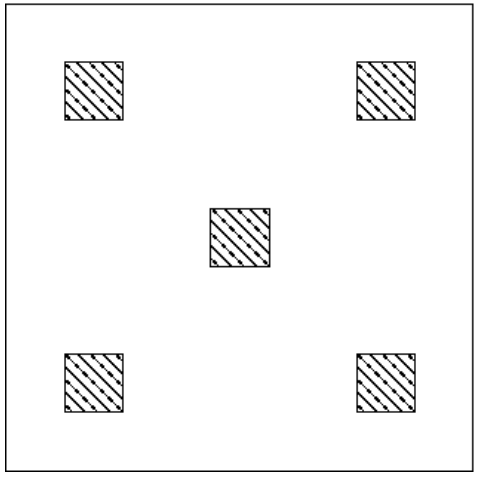
approx. discretization error

Semilinear Equation

$$\min \frac{1}{2} \|y - y_d\|^2_{L^2(\Omega\times(0,T))} + \frac{\alpha}{2} \|u\|^2_{L^2(0,T;\mathbb{R}^5)}$$

subject to

$$\begin{aligned} y_t - \varepsilon^2 \Delta y &= y(y-a)(1-y) + u && \text{in } \Omega \times (0,T) \\ \partial_\nu y &= 0 && \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= y_0 && \text{in } \Omega \end{aligned}$$



and

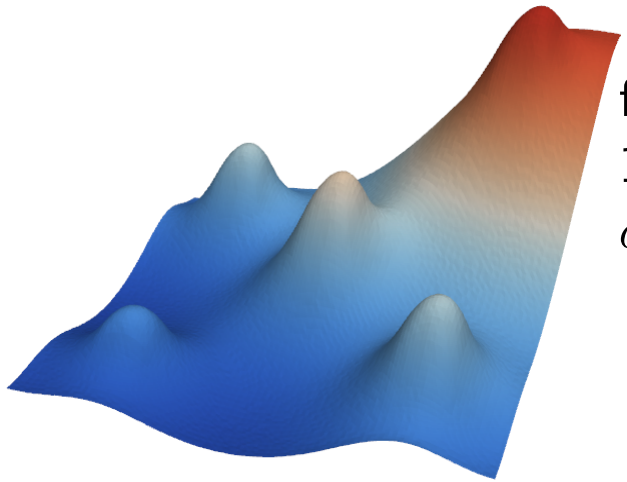
$$\Omega = [0,1]\times[0,1], \quad T = 10, \quad a = 0.1, \quad \alpha = 10^{-5}, \quad \varepsilon = 0.15$$

$$y_d(x,t) = \frac{1}{1 + e^{\left((\|x\| - \frac{1}{3})\cdot\frac{1}{\varepsilon\sqrt{2}} - t\right)}}, \quad y_0(x) = y_d(x,0).$$

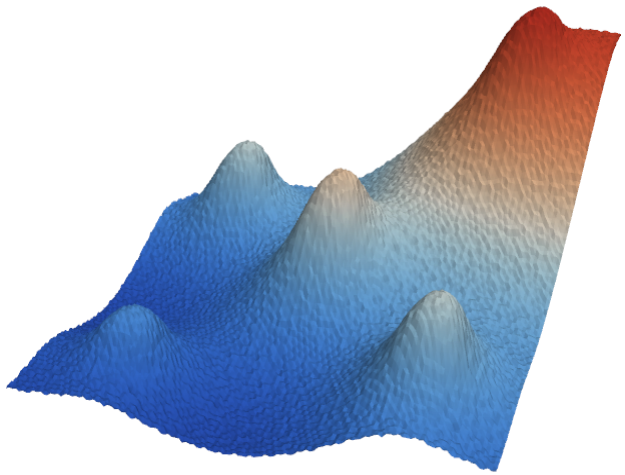
Semilinear Equation

Reconstructed states

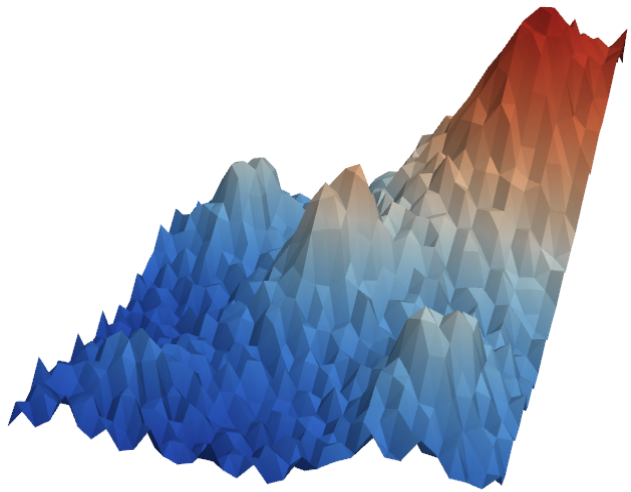
at $t = 0.8$,
50 iterations



factor 56,
1.14 bits/vertex
$\delta = 5 \cdot 10^{-4}$



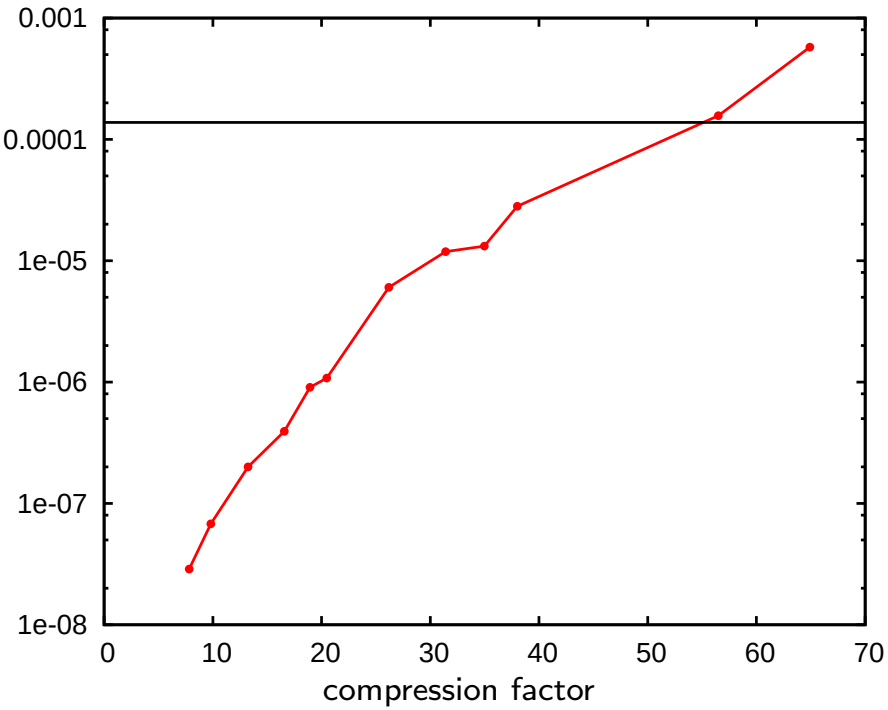
factor 96, 0.67 bits/vertex
$\delta = 5 \cdot 10^{-3}$



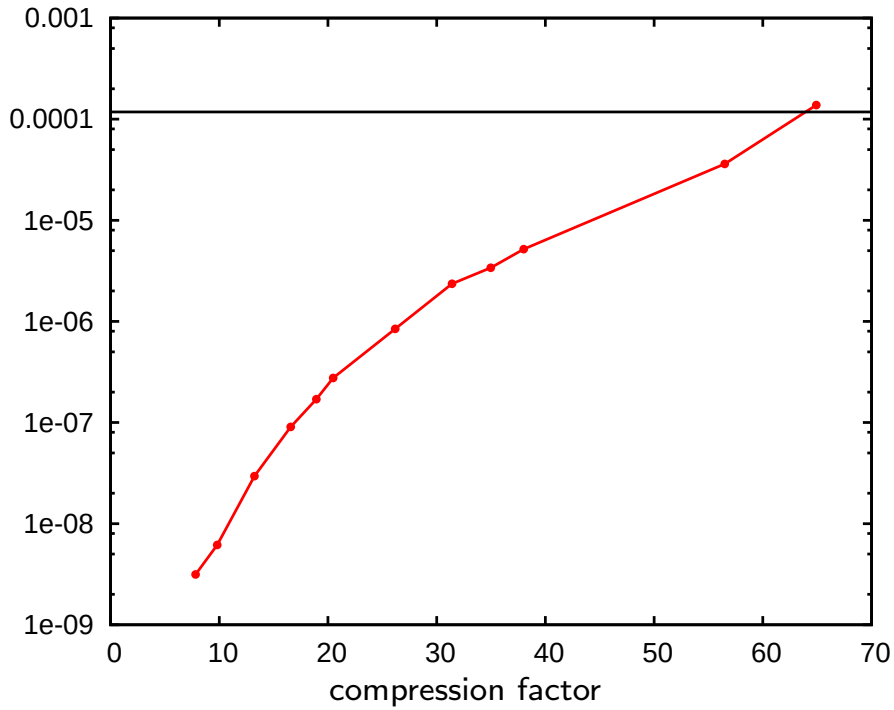
factor 401, 0.16 bits/vertex
$\delta = 5 \cdot 10^{-2}$

Semilinear Equation

relative L^∞ error reduced gradient



relative L^∞ error control

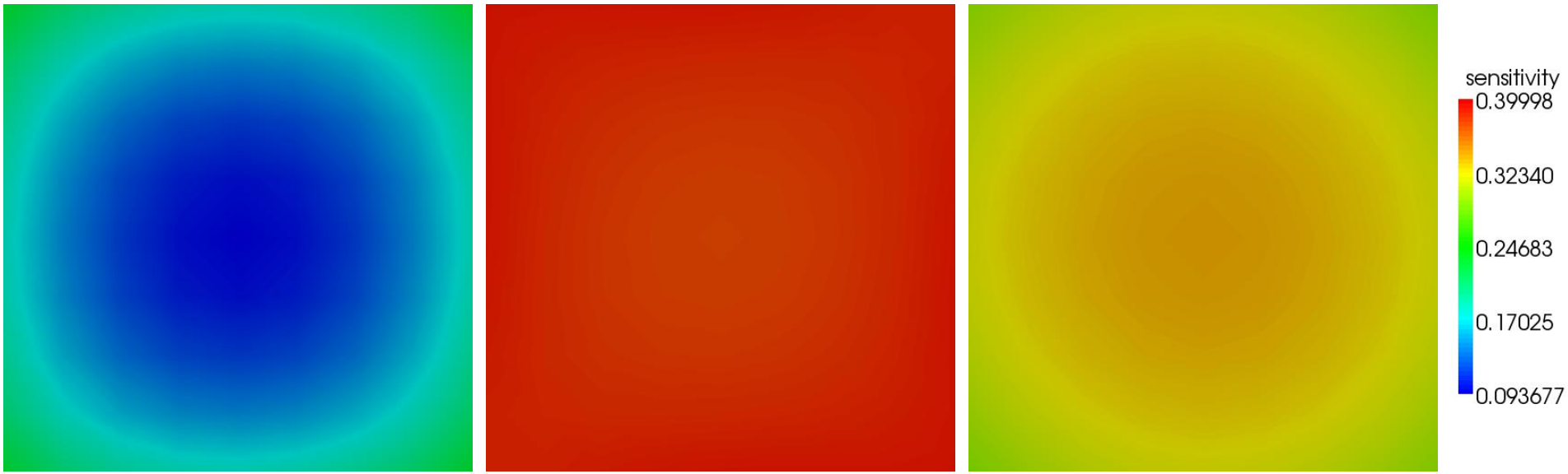


Outlook

Adaptive Quantization

Sensitivity of the reduced gradient error norm

$$\frac{\partial \left\| \delta \hat{j}(u) \right\|_{L^2}^2}{\partial \delta y} = 2 J_{yy}^{\star} c_y^{-1} c_u c_u^{\star} c_y^{-\star} J_{yy} \delta y$$



non-constant in space and time

- choose spatio-temporally varying quantization levels

Conclusion

Trajectory Compression

- Reduced gradient inexact – lossy compression a viable approach
- Considerable storage reduction
- Computationally inexpensive
- Easy to use

Conclusion

Trajectory Compression

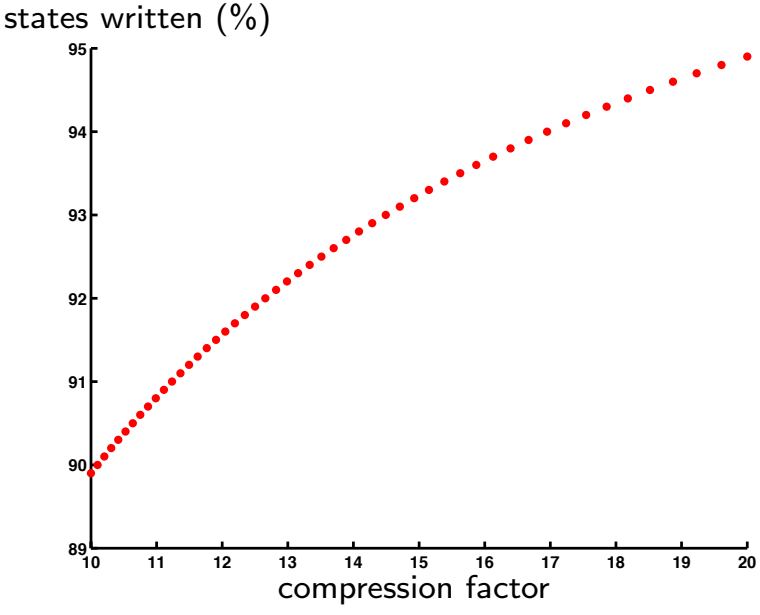
- Reduced gradient inexact – lossy compression a viable approach
- Considerable storage reduction
- Computationally inexpensive
- Easy to use

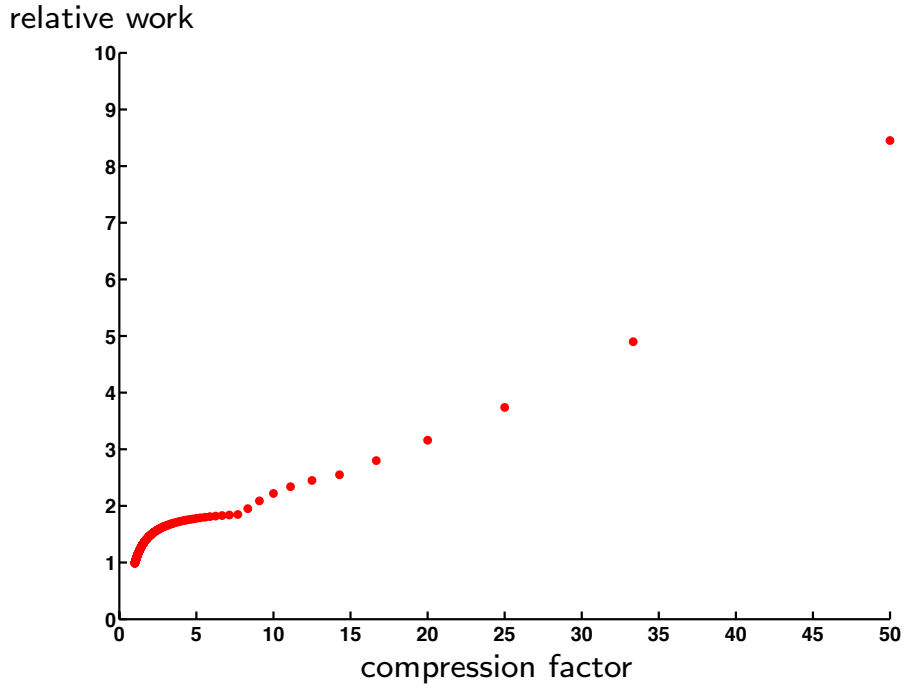
Thank you for your attention!

Checkpointing [Griewank, Walther, Stumm]

- save state values at selected timesteps
- recompute parts of trajectory

optimal checkpoint distribution for fixed number of timesteps, fixed number of checkpoints





Tradeoff: computational work – size

Memory access: Re-use of checkpoints, no significant reduction of write/read counts