Simulation based Optimization

Eldad Haber

August 18, 2008

Goals

- PDE optimization problems can be very involved.
- Try to explain the essence and possible pitfalls
- Encourage you to get into this *cool!* field
- Give some simple software to demonstrate these concepts

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Outline

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Matlab Code

Simulation and Optimization

The (continuous) problem:

$$\begin{aligned} \min \quad & \mathcal{J}(y,u) \\ \text{subject to} \quad & c(y,u) = 0 \end{aligned}$$

$u \in \mathcal{U}$ model - control
$y \in \mathcal{Y}$ field - state
$\mathcal{J} : [\mathcal{U} \times \mathcal{Y}] \to \mathcal{R}^1$
$c : [\mathcal{U} \times \mathcal{Y}] \to \widehat{\mathcal{Y}}$

Simulation and Optimization

The (discrete) problem:

$$\begin{array}{rl} \text{min} & \mathcal{J}(y,u) \\ \text{subject to} & c(y,u)=0 \end{array}$$

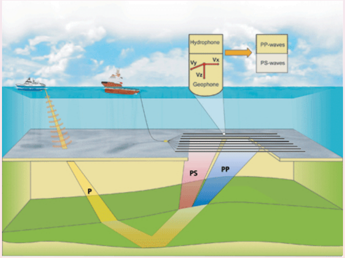
$u \in \mathcal{R}^n$ model - control
$y \in \mathcal{R}^m$ field - state
$\mathcal{J} : \mathcal{R}^{m+n} \to \mathcal{R}^1$
$c : \mathcal{R}^{m+n} \to \mathcal{R}^m$

Example I

Seismic inversion Clerbout 2000

$$\min \qquad \mathcal{J} = \frac{1}{2}\sum_i \|Q_j y_j - d\|^2 + \frac{\alpha}{2}\|Lu\|^2$$

$$\text{s.t.} \qquad c(y_j, u) = \Delta_h y_j + k^2 u \odot y_j = 0 \qquad j = 1, \ldots n_s$$

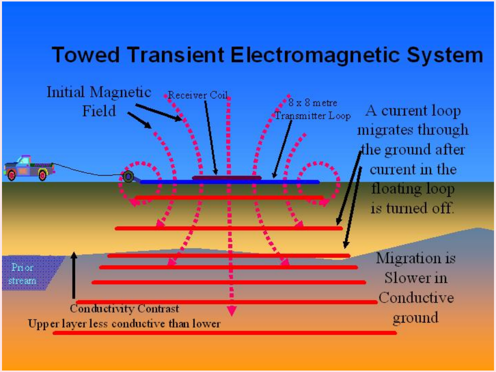


Example II

Electromagnetic inversion Newman 1996

$$\min \quad \mathcal{J} = \frac{1}{2}\sum_j \|Q_j y_j - d\|^2 + \frac{\alpha}{2}\|Lu\|^2$$

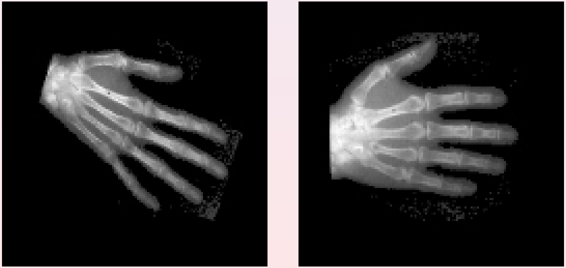
$$\text{s.t.} \quad c(y_j, u) = (\nabla \times \mu^{-1}\nabla\times)_h y_j + i\omega S(u) y_j = 0 \quad j = 1, \dots, n_s$$



Example III

Image Processing - transprot Modersitzki 2003

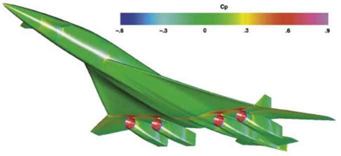
$$
\begin{array}{rl}
\min & \mathcal{J} = \frac{1}{2}\|y(T,x) - d(x)\|^2 + \alpha S(u) \\
\text{subject to} & y_t + u^\top \nabla y = 0 \qquad y(0,x) = y_0(x)
\end{array}
$$



Example IV

Shape Optimization Haslinger & Makinen 2003

$$\begin{aligned} \min \quad & \mathcal{J} = g(y) \\ \text{subject to} \quad & c(y,u) = \Delta_h y - f(u) = 0 \end{aligned}$$



Some Historical Prespective

Optimization with O/PDE constraint is common practice in many applications for many years

- Geophysical inversion for conductivity (Schlumberger 1912)
- **Other fields:** Flow design, VLSI, trajectory planning, chemical reaction control, ... (starting in the 30's and on)

However,

- better computer architecture $\longrightarrow$ larger simulations
- development in numerical PDE's $\longrightarrow$ complex models

New optimization algorithms are needed

Some Historical Prespective

Optimization with O/PDE constraint is common practice in many applications for many years

- Geophysical inversion for conductivity (Schlumberger 1912)
- **Other fields:** Flow design, VLSI, trajectory planning, chemical reaction control, ... (starting in the 30's and on)

However,

- better computer architecture $\longrightarrow$ larger simulations
- development in numerical PDE's $\longrightarrow$ complex models

New optimization algorithms are needed

Some Historical Prespective

Optimization with O/PDE constraint is common practice in many applications for many years

- Geophysical inversion for conductivity (Schlumberger 1912)
- **Other fields:** Flow design, VLSI, trajectory planning, chemical reaction control, ... (starting in the 30's and on)

However,

- better computer architecture $\rightarrow$ larger simulations
- development in numerical PDE's $\rightarrow$ complex models

New optimization algorithms are needed

Some Historical Prespective

Optimization with O/PDE constraint is common practice in many applications for many years

- Geophysical inversion for conductivity (Schlumberger 1912)
- **Other fields:** Flow design, VLSI, trajectory planning, chemical reaction control, ... (starting in the 30's and on)

However,

- better computer architecture $\rightarrow$ larger simulations
- development in numerical PDE's $\rightarrow$ complex models

New optimization algorithms are needed

Before we do anything

All you got to do is think Pooh Bear

Our framework: Discretize-Optimize

$$\min \ \mathcal{J}(y,u) \qquad \text{s.t} \ \ c(y,u)=0$$

Optimize-Discretize: *Can yield inconsistent gradients of the objective functionals. The approximate gradient obtained in this way is not a true gradient of anything–not of the continuous functional nor of the discrete functional.*

Discretize-Optimize *Requires to differentiate computational facilitators such as turbulence models, shock capturing devices or outflow boundary treatment.*

M. Gunzburger

Want to use the wealth of optimization algorithms

Simulation and Optimization

- Need to discretize the PDE (constraint)
- Parameters change - modeling need to be flexible
- Need to optimize - derivatives

Discretizing $c(y, u) = 0$ - difficulties

Stability with respect to parameters

$$c(y, u) = y_t - u y_{xx}$$

Explicit vs Implicit

Explicit:

$$c_h(y_h, u_h) = y_h^{n+1} - y_h^n - u_h \odot \frac{\delta t}{\delta x^2} L y_h^n = 0$$

Discretizing $c(y, u) = 0$ - difficulties

Stability with respect to parameters

- Stability requires $u_h \delta t \approx \delta x^2$
- do not know u $\rightarrow$ hard to guarantee stability.
- Code has to make sure discretization is compatible
- Possible solution: implicit methods are unconditionally stable

Discretizing $c(y, u) = 0$ - difficulties

Stability with respect to parameters

$$c(y, u) = y_t - u y_{xx}$$

Explicit vs Implicit

Implicit:

$$c_h(y_h, u_h) = y_h^{n+1} - y_h^n - u_h \odot \frac{\delta t}{\delta x^2} L y_h^{n+1} = 0$$

No free lunch, need to invert a matrix

Discretizing $c(y,u) = 0$ - difficulties

Differentiability of the discretization

$$c(y,u) = \epsilon y_{xx} + u y_x = 0$$

Common discretization, upwind

$$\frac{\epsilon}{h^2}(y_{j+1} - 2y_j + y_{j-1}) +$$
$$\frac{1}{h}\left(\max(u_j,0)(y_j - y_{j-1}) + \min(u_j,0)(y_{j+1} - y_j)\right) = 0$$

Discretizing $c(y,u)=0$ - difficulties

The continuous problem is continuously differentiable w.r.t u

$$\epsilon y_{xx} + u y_x = 0$$

The discrete problem is not differentiable w.r.t u_h

$$\frac{\epsilon}{h^2}(y_{j+1} - 2y_j + y_{j-1}) +$$
$$\frac{1}{h}\left(\max(u_j,0)(y_j - y_{j-1}) + \min(u_j,0)(y_{j+1} - y_j)\right) = 0$$

Even more difficult for flux limiters

Discretizing $c(y, u) = 0$ - difficulties

The continuous problem is continuously differentiable w.r.t u but discrete problem is not

$$\epsilon y_{xx} + u y_x = 0$$

No magic solution for this one - can pose real difficulty for the DO approach

Discretizing $c(y, u) = 0$ - difficulties

Nonlinearity of the discretization

"the mother of all elliptic problems" Dendy 1991

$$-\nabla \cdot (u \nabla y) = q$$

Finite volume discretization

$$A(u_h) y_h = \overbrace{D^\top}^{-\nabla\cdot} \underbrace{\mathrm{diag}\,(N(u_h))}_{u} \overbrace{D}^{\nabla} y_h = q_h$$

where $N(u_h) = (A_v u_h^{-1})^{-1}$ harmonic averaging

The continuous problem is bilinear but discrete problem is more nonlinear.

Discretizing $c(y, u) = 0$ - difficulties

Nonlinearity of the discretization

"the mother of all elliptic problems" Dendy 1991

$$-\nabla \cdot (u \nabla y) = q$$

Finite volume discretization

$$A(u_h) y_h = \overbrace{D^\top}^{-\nabla\cdot} \underbrace{\mathrm{diag}\,(N(u_h))}_{u} \overbrace{D}^{\nabla} y_h = q_h$$

where $N(u_h) = (A_v u_h^{-1})^{-1}$ harmonic averaging

The continuous problem is bilinear but discrete problem is more nonlinear.

Discretizing $c(y,u)=0$ - difficulties

Nonlinearity of the discretization

$$-\nabla\cdot\,(u\nabla y)=q$$

Finite volume discretization

$$A(u_h)y_h=\overbrace{D^\top}^{-\nabla\cdot}\ \underbrace{\mathrm{diag}\left(N(u_h)\right)}_{u}\ \overbrace{D}^{\nabla}\ y_h=q_h$$

Differentiate the discrete approximation rather than the continuous one

Before we solve

- PDE optimization problems are different because PDE's are different
- To make progress need to classify them. Use similar tools for similar problems
- Need good model problems to experiment with

Discretization - summary

Classify PDE's using 2 categories

- PDE's that are smooth enough such that the DO approach works well
- PDE's that require special attention in their discretization, need OD

Although we look at the PDE through the discretization these properties are intrinsic to the PDE itself

Discretization - summary

Classify PDE's using 2 categories

- Smooth PDE's such that the DO approach works well
    - Elliptic problems
    - Parabolic problems
    - Smooth hyperbolic problems
    - Some nonlinear problems
- PDE's require special attention in their discretization, need OD
    - Hyperbolic problems with nonsmooth initial data
    - Nonlinear problems with shocks
    - Other Nonlinear problems e.g, Eikonal and alike

Accuracy issues

- For many problems, constraint must be taken seriously (physics) but the optimization less so (noise, regularization)
- In many cases the control-model change little after the first reduction of the objective function

Example:

$$\begin{array}{rl} \min & \|u-b\|^2 + \alpha TV_\epsilon(u) \\ \text{s.t} & \nabla \cdot u \nabla y = q \end{array}$$

where

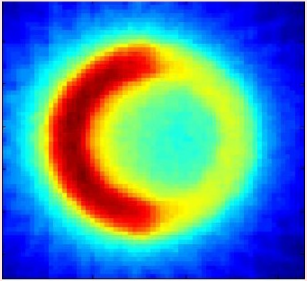
$$TV_\epsilon(t) = \begin{cases} \frac{1}{2\epsilon} t^2 + \frac{\epsilon}{2} & |t| \le \epsilon \\ |t| & |t| > \epsilon \end{cases}$$

Accuracy issues

Example:

$$\begin{aligned} \min \quad & \|u-b\|^2 + \alpha TV_\epsilon(u) \\ \text{s.t} \quad & \nabla \cdot u \nabla y = q \end{aligned}$$

$\epsilon = 10^0$

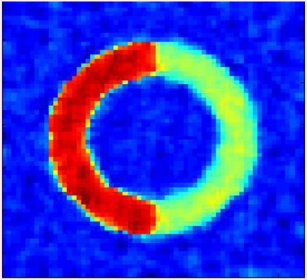


Accuracy issues

Example:

$$\begin{array}{rl} \min & \|u-b\|^2 + \alpha TV_\epsilon(u) \\ \text{s.t} & \nabla \cdot u \nabla y = q \end{array}$$

$\epsilon = 10^{-1}$

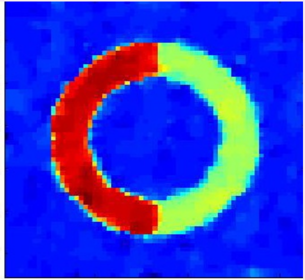


Accuracy issues

Example:

$$\begin{array}{rl} \min & \|u-b\|^2 + \alpha TV_\epsilon(u) \\ \text{s.t} & \nabla \cdot u \nabla y = q \end{array}$$

$\epsilon = 10^{-2}$



Optimization

Can we build it? Yes we can! Bob the builder

Solving the optimization problem

Constrained approach, solve

$$\begin{aligned} \text{min} \qquad & \mathcal{J}(y,u) \\ \text{subject to} \qquad & c(y,u) = 0 \end{aligned}$$

Unconstrained approach, eliminate y to obtain

$$\text{min} \qquad \mathcal{J}(y(u),u)$$

Constrained vs. unconstrained

Example: $c(y,u) = A(u)y - q = 0$
Constrained approach,

$$\begin{array}{rl} \min & \mathcal{J}(y,u) \\ \text{subject to} & A(u)y = q \end{array}$$

Unconstrained approach,

$$\min \quad \mathcal{J}(A(u)^{-1}q, u)$$

- Invertibility of $A(u)$
- Cost of evaluating the ObjFun.

Constrained vs. unconstrained

Example: $c(y,u) = A(u)y - q = 0$
Constrained approach,

$$\begin{aligned} \min & \quad \mathcal{J}(y,u) \\ \text{subject to} & \quad A(u)y = q \end{aligned}$$

Unconstrained approach,

$$\min \quad \mathcal{J}(A(u)^{-1}q, u)$$

- Invertibility of $A(u)$
- Cost of evaluating the ObjFun.

Constrained vs Unconstrained

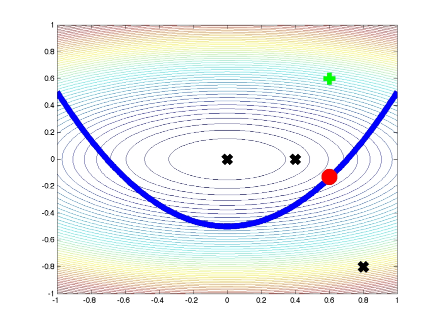
Constrained approach,

- Saddle point problem
- Algorithmically hard
- No need to solve the constraints until the end

Unconstrained approach

- Simple from an optimization standpoint
- Need to solve the constraint equation PDE
- Becomes even messier for nonlinear PDE's
- But: always feasible!!!

Constrained vs Unconstrained



Sequential Quadratic Programming

The Lagrangian

$$\mathcal{L} = \mathcal{J}(y,u) + \lambda^\top Mc(y,u)$$

where

$$\lambda^\top Mc(y,u) \approx \int_\Omega \lambda(x)c(y(x),u(x))\,dx$$

Differentiate to obtain the Euler Lagrange equations (Assume $M = I$)

$$\begin{aligned} \text{adjoint} \qquad & \mathcal{J}_y + c_y^\top \lambda = 0 \\ \text{state} \qquad & \mathcal{J}_u + c_u^\top \lambda = 0 \\ \text{constraint} \qquad & c(y,u) = 0 \end{aligned}$$

Computing Jacobians

- Need to compute c_y, c_u
- In many cases c_y available (used for the forward)
- Need to compute c_u, calculus with matrices helps
- In some cases c_y not used for the forward

Jacobians, example I:Hydrology, electromagnetics

$$c(y,u) = A(u)y - q = D^\top \mathrm{diag}((A_v u^{-1})^{-1}) D y - q$$

Then

$$\begin{aligned} c_y &= A(u) \\ c_u &= \frac{\partial}{\partial u}\left[D^\top \mathrm{diag}((A_v u^{-1})^{-1}) D y \right] \end{aligned}$$

Note that

$$D^\top \mathrm{diag}((A_v u^{-1})^{-1}) D y = D^\top \mathrm{diag}(Dy)\ (A_v u^{-1})^{-1}$$

therefore

$$c_u = D^\top \mathrm{diag}(Dy)\ \mathrm{diag}((A_v u^{-1})^{-2}) A_v \mathrm{diag}(u^{-2})$$

Jacobians, example II : CFD

NS equations

$$\Delta_h y + M(y)y + \nabla_h p = u$$
$$\nabla_h \cdot y_k = 0$$

Where $M(y) \approx \nabla y$
Typical solution through fixed point iteration [Elman, Silvester, Wathen]

$$\Delta_h y_k + M(y_{k-1})y_k + \nabla_h p = u$$
$$\nabla_h \cdot y_k = 0$$

Thus to compute $c(y)$ need extra calculation

Jacobians, example II : CFD

In general

$$c(y,u) = 0$$

Use some iteration to solve (not Newton's method)
From an optimization theory we need the Jacobians c_y, c_p of the constraint otherwise cannot guarantee convergence

Open Question: Can we get away with less?

Two alternative viewpoints

$$\begin{aligned} \text{adjoint} \qquad & \mathcal{J}_y + c_y^\top \lambda = 0 \\ \text{state} \qquad & \mathcal{J}_u + c_u^\top \lambda = 0 \\ \text{constraint} \qquad & c(y, u) = 0 \end{aligned}$$

A system of nonlinear PDE's
use PDE techniques
(MG, FAS, ...)

Necessary conditions
use optimization framework
(reduce Hessian ...)

MG(linear)
MGOPT [Luis & Nash]

Two alternative viewpoints

A system of nonlinear PDE's
use PDE techniques
(MG, FAS, ...)

Necessary conditions
use optimization framework
(reduce Hessian ...)

Our approach:
Use PDE techniques as solvers
Use optimization methods for a guide

Two alternative viewpoints

A system of nonlinear PDE's
use PDE techniques
(MG, FAS, ...)

Necessary conditions
use optimization framework
(reduce Hessian ...)

Our approach:
Use PDE techniques as solvers
Use optimization methods for a guide

Solving the Euler Lagrange equations

$$\begin{aligned} \text{adjoint} \qquad & \mathcal{J}_y + c_y^\top \lambda = 0 \\ \text{state} \qquad & \mathcal{J}_u + c_u^\top \lambda = 0 \\ \text{constraint} \qquad & c(y,u) = 0 \end{aligned}$$

Approximate the Hessian and solve at each iteration the KKT system

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Solving the Euler Lagrange equations

In many applications approximate the Hessian by

$$\begin{pmatrix} \mathcal{J}_{yy} & \mathbf{O} & c_y^\top \\ \mathbf{O} & \mathcal{J}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Gauss-Newton SQP [Bock 89]

If $\mathcal{J}_{yy}$ and $\mathcal{J}_{uu}$ are positive semidefinite then the reduced Hessian is likely to be SPD.

Solving the KKT system

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & O \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

- Direct methods are (almost) out of the question!
- Multigrid methods for the KKT system
- The reduced Hessian
- Preconditioners

Solving the KKT system - multigrid

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & O \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

- Multigrid is a good tool to study the problem
- May use other techniques at the end
- Learn about the discretization/solver

Solving the KKT system - multigrid

Ascher & H. 2000, Kunish & Borzi 2003

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & O \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

- Check ellipticity of the continuous problem
- Check h-ellipticity of the discrete problem

Multigrid h-ellipticity

Look at the symbol Ta'asan

$$\widehat{H}(\theta) = \begin{pmatrix} \widehat{\mathcal{L}}_{yy} & & \widehat{c}_y^* \\ & \widehat{\mathcal{L}}_{uu} & \widehat{c}_u^* \\ \widehat{c}_y & \widehat{c}_u & 0 \end{pmatrix}$$

Compute the determinant

$$|\det(H)(\theta)| = \widehat{\mathcal{L}}_{yy}\widehat{c}_u^*\widehat{c}_u + \widehat{\mathcal{L}}_{uu}\widehat{c}_y^*\widehat{c}_y$$

Look at high frequencies

Example

Load problem

$$\min \; \frac{1}{2}\|y-d\|^2 + \frac{\alpha}{2}\|Lu\|^2 \quad \text{s.t } \Delta y - u = 0$$

$$\widehat{H}(\theta) = \begin{pmatrix} 1 & & \widehat{\Delta}_h \\ & \alpha\widehat{L} & 1 \\ \widehat{\Delta} & 1 & 0 \end{pmatrix}$$

Compute the determinant of the symbol ($\widehat{\Delta}_h = h^{-2}2(\cos(\theta_1) + cos(\theta_2) - 2)$)

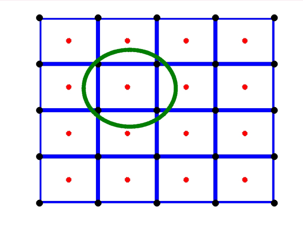
$$|\det(H)(\theta)| = 1 + \alpha\widehat{L}\widehat{\Delta}_h^2$$

Look at high frequencies

Solving the KKT system - multigrid

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & O \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Box smoothing - solve the equation locally



Solving the KKT system - multigrid

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & O \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Need:

- smoother - box smoothing, others?(in progress)
- coarse grid approximation
- solution on the coarsest grid (may not be so coarse)

Solving the KKT system - multigrid

- Case by case development
- Hard to generalize, even when BC change
- May worth the effort if the same type of problem is repeatedly solved

Solving the KKT system - The reduced Hessian

Nocedal & Wright 1999

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathbf{O} & c_y^\top \\ \mathbf{O} & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

- Eliminate s_y
  $c_y s_y + c_u s_u = \ldots$
- Eliminate s_λ
  $\mathcal{L}_{yy} s_u + c_u^\top s_\lambda = \ldots$
- Obtain an equation for s_u

$$H_r s_u = \underbrace{\left(c_u^\top c_y^{-\top} \mathcal{L}_{yy} c_y^{-1} c_u + \mathcal{L}_{uu} \right)}_{\text{the reduced Hessian}} s_u = \text{rhs}$$

The reduced Hessian in Fourier space

Use LFA to study the properties of the reduced Hessian.
Load problem

$$\min \ \frac{1}{2}\|y-d\|^2 + \frac{\alpha}{2}\|Lu\|^2 \quad \text{s.t } \Delta y - u = 0$$

$$\widehat{H}(\theta) = \begin{pmatrix} 1 & & \widehat{\Delta}_h \\ & \alpha\widehat{L} & 1 \\ \widehat{\Delta} & 1 & 0 \end{pmatrix}$$

The symbol of the reduced Hessian ($\widehat{\Delta}_h = h^{-2}2(\cos(\theta_1) + cos(\theta_2) - 2)$)

$$\widehat{\Delta}_h^{-2} + \alpha\widehat{L}$$

Very unstable for small α

More on the reduced Hessian method

$$H_r s_u = \left(c_u^\top c_y^{-\top} \mathcal{L}_{yy} c_y^{-1} c_u + \mathcal{L}_{uu}\right) s_u = \text{rhs}$$

- For QP with linear constraints the reduced Hessian is equivalent to the Hessian of the unconstrained approach
- The reduced Hessian represents an integro-differential equation
- Efficient solvers for the reduced Hessian is an open question, recent work [Biros & Dugan]

Even more on the reduced Hessian method

The reduced Hessian can be viewed as a block factorization of the (permuted) KKT system H. & Ascher 2001, Biros & Gahttas 2005, Dollar & Wathen 2006

$$\begin{pmatrix} c_y & \mathbf{O} & c_u \\ \mathcal{L}_{yy} & c_y^\top & \mathbf{O} \\ \mathbf{O} & c_u & \mathcal{L}_{uu} \end{pmatrix}^{-1} =$$

$$\begin{pmatrix} c_y^{-1} & \mathbf{O} & -JH_r^{-1} \\ \mathbf{O} & c_y^{-\top} & -c_y^{-\top}JH_r^{-1} \\ \mathbf{O} & \mathbf{O} & H_r^{-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ c_y^{-1} & \mathbf{I} & \mathbf{O} \\ -J^\top c_y^{-1} & -J^\top & \mathbf{I} \end{pmatrix}$$

$$J = c_y^{-1} c_u$$

$$H_r = J^\top J + \mathcal{L}_{uu}$$

Solving the KKT system - iterative methods and preconditioners

Solve

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Using some Krylov method (MINRES, SYMQMR, GMRES, ...)

- Indefinite
- Highly ill-conditioned
- A must: Preconditioner

Many of the preconditioners developed for general optimization problems are not useful

Solving the KKT system - iterative methods and preconditioners

Solve

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Using some Krylov method (MINRES, SYMQMR, GMRES, ...)

- Indefinite
- Highly ill-conditioned
- A must: Preconditioner

Many of the preconditioners developed for general optimization problems are not useful

Solving the KKT system - iterative methods and preconditioners

Solve

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \text{rhs}$$

Using some Krylov method (MINRES, SYMQMR, GMRES, ...)

- Indefinite
- Highly ill-conditioned
- A must: Preconditioner

Many of the preconditioners developed for general optimization problems are not useful

Solving the KKT system - iterative methods and preconditioners

$$\begin{pmatrix} \mathcal{L}_{yy} & \mathcal{L}_{yu} & c_y^\top \\ \mathcal{L}_{yu}^\top & \mathcal{L}_{uu} & c_u^\top \\ c_y & c_u & \mathbf{O} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \\ s_\lambda \end{pmatrix} = \mathrm{rhs}$$

Preconditioners based on the approximate reduced Hessian method H. & Ascher 2001, Biros & Ghattas 2005

Preconditioners based on the reduced Hessian method

$$\begin{pmatrix} c_y & \mathbf{O} & c_u \\ \mathcal{L}_{yy} & c_y^\top & \mathbf{O} \\ \mathbf{O} & c_u & \mathcal{L}_{uu} \end{pmatrix}^{-1} \approx$$

$$\begin{pmatrix} \widehat{c}_y^{-1} & \mathbf{O} & -\widehat{J}H_r^{-1} \\ \mathbf{O} & \widehat{c}_y^{-\top} & -\widehat{c}_y^{-\top}\widehat{J}\widehat{H}_r^{-1} \\ \mathbf{O} & \mathbf{O} & \widehat{H}_r^{-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ \widehat{c}_y^{-1} & \mathbf{I} & \mathbf{O} \\ -\widehat{J}^\top \widehat{c}_y^{-1} & -\widehat{J}^\top & \mathbf{I} \end{pmatrix}$$

$$\widehat{J} = \widehat{c}_y^{-1} c_u$$

$$\widehat{H}_r = ??$$

Preconditioners based on the reduced Hessian method

$$\begin{pmatrix} c_y & \mathbf{O} & c_u \\ \mathcal{L}_{yy} & c_y^\top & \mathbf{O} \\ \mathbf{O} & c_u & \mathcal{L}_{uu} \end{pmatrix}^{-1} \approx \begin{pmatrix} \widehat{c}_y^{-1} & \mathbf{O} & -\widehat{J}H_r^{-1} \\ \mathbf{O} & \widehat{c}_y^{-\top} & -\widehat{c}_y^{-\top}\widehat{J}\widehat{H}_r^{-1} \\ \mathbf{O} & \mathbf{O} & \widehat{H}_r^{-1} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ \widehat{c}_y^{-1} & \mathbf{I} & \mathbf{O} \\ -\widehat{J}^\top \widehat{c}_y^{-1} & -\widehat{J}^\top & \mathbf{I} \end{pmatrix}$$

Approximating c_y and H_r

- $\widehat{c}_y$ - standard PDE approximation
- $\widehat{H}_r$ - BFGS, other QN, approximate inverse, ...
- Can prove mesh independence under some assumptions

Other Preconditioners

Other approaches

- Domain Decomposition, [Heinkenschloss 02]
- Augmented Lagrangian, [Greif & Golub 03]
- Schur complement based
- See excellent review paper by Benzi
  Everything you wanted to know about KKT systems but was afraid to ask

No magic bullet, application dependent (as they should be!)

Taking a step

$$\min \mathcal{J}(y,u) \quad \text{s.t}\; c(y,u) = 0$$

Guess u_0, y_0
while not converge

- Evaluate $\mathcal{J}_k, c_k$, $\nabla\mathcal{L}_k, c_y, c_u$ and an approximation to the Hessian (the KKT system)
- Approximately solve the KKT system for a step
- Take a (guarded) step
- Check if need to project to the constraint

Questions

while not converge

- Evaluate $\mathcal{J}_k, c_k$, $\nabla\mathcal{L}_k, c_y, c_u$ and an approximation to the Hessian (the KKT system)
  How accurate should the Hessian/Jacobian be?
- Approximately solve the KKT system for a step
  To what tolerance?
- Take a (guarded) step
  How should we judicially pick a step?
- Check if need to project to the constraint
  why and when should we project?

Questions

while not converge

- Evaluate $\mathcal{J}_k, c_k$, $\nabla\mathcal{L}_k, c_y, c_u$ and an approximation to the Hessian (the KKT system)
  How accurate should the Hessian/Jacobian be?
- Approximately solve the KKT system for a step
  To what tolerance?
- Take a (guarded) step
  How should we judicially pick a step?
- Check if need to project to the constraint
  why and when should we project?

Questions

while not converge

- Evaluate $\mathcal{J}_k, c_k$, $\nabla\mathcal{L}_k, c_y, c_u$ and an approximation to the Hessian (the KKT system)
  How accurate should the Hessian/Jacobian be?
- Approximately solve the KKT system for a step
  To what tolerance?
- Take a (guarded) step
  How should we judicially pick a step?
- Check if need to project to the constraint
  why and when should we project?

Questions

while not converge

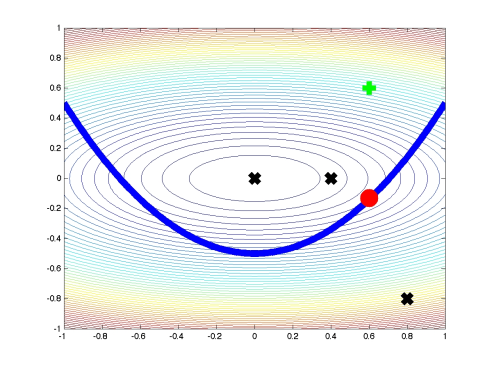
- Evaluate $\mathcal{J}_k, c_k$, $\nabla\mathcal{L}_k, c_y, c_u$ and an approximation to the Hessian (the KKT system)
  How accurate should the Hessian/Jacobian be?
- Approximately solve the KKT system for a step
  To what tolerance?
- Take a (guarded) step
  How should we judicially pick a step?
- Check if need to project to the constraint
  why and when should we project?

How well should we solve the KKT system?

- treat the problem as a system of nonlinear equations we can use inexact Newton's theory - ignore optimization aspects
- for traditional SQP algorithms require accurate solutions
- Can we use SQP with inaccurate solution of the sub-problem?

  Leibfritz & Sachs 1999, Heinkenschloss & Vicente 2001
- Recent work by Curtis Nocedal and Bird on inexact SQP methods, based on a penalty function

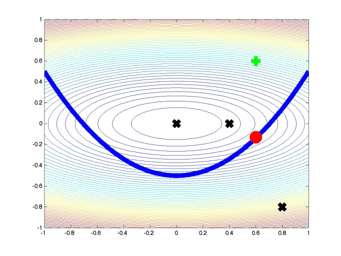
Choosing a step



The dilemma

- Should I decrease the Objective?
- Should I become more feasible?

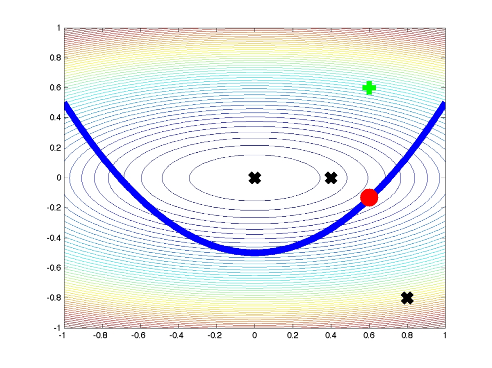
Choosing a step



merit function approach: $\mathcal{L}_\mu = f(y,u) + \mu h(c(y,u))$

- Use the L_1 or L_2 merit functions
- Disadvantage - need an estimate of the Lagrange multipliers

Choosing a step



Filter Fletcher & Leyffer 2002

- either reduce the objective or
- improve feasibility
- No need for Lagrange multipliers

Projecting back to the constraint

- In most cases feasibility is much more important than optimality
- Project the solution when getting close or before termination
- Can help with convergence (secondary correction)

Projecting back to the constraint

- In most cases feasibility is much more important than optimality
- Project the solution when getting close or before termination
- Can help with convergence (secondary correction)

Projecting back to the constraint

- In most cases feasibility is much more important than optimality
- Project the solution when getting close or before termination
- Can help with convergence (secondary correction)

Projecting back to the constraint - beyond optimization

- Accuracy of the optimization can be low
- Accuracy of the PDE should be high
- When should we project?

Multilevel

- Multilevel approach is computational effective
- In many cases, avoid local minima
- Help choosing parameters (e.g regularization, interior point)
- Hard to prove

Grid Sequencing

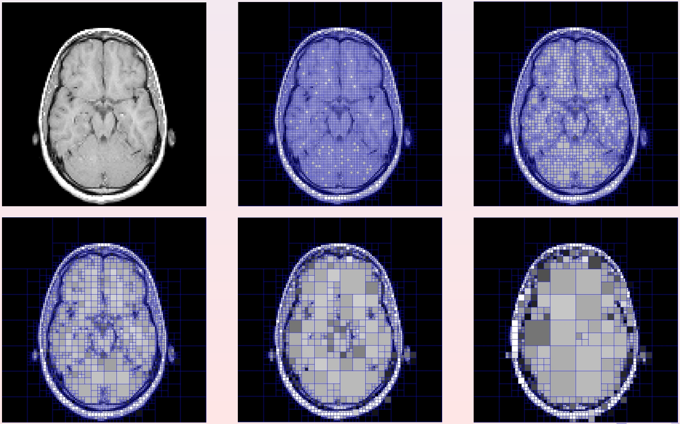
The problems we solve have an underline continuous structure.
Use this structure for continuation

Main idea: *Solution of the problem on a coarse grid can approximate the problem on a fine grid.*

Use coarse grids to evaluate parameters within the optimization. Moŕe , Burger, Ascher & H., H. & Modersitzki, H., H. & Benzi

Adaptive Multilevel Grid Sequencing

- Rather than refine everywhere, refine only where needed H., Heldman & Ascher [07], Bungrath [08]
- Requires data structures, discretization techniques, refinement techniques
- Can save an order of magnitude in calculation

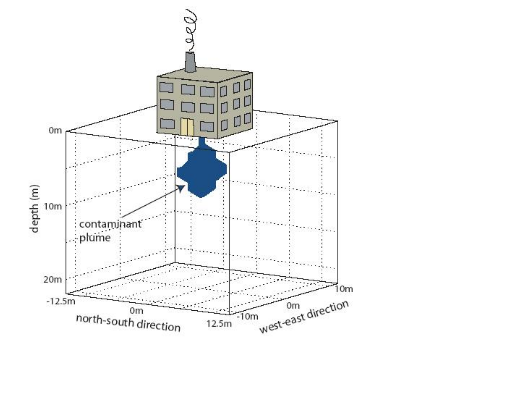


Examples

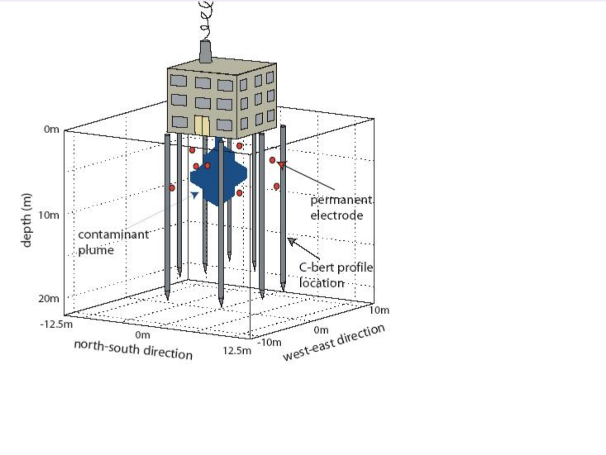
And this is how its really done Dora the explorer

Application: Impedance Tomography

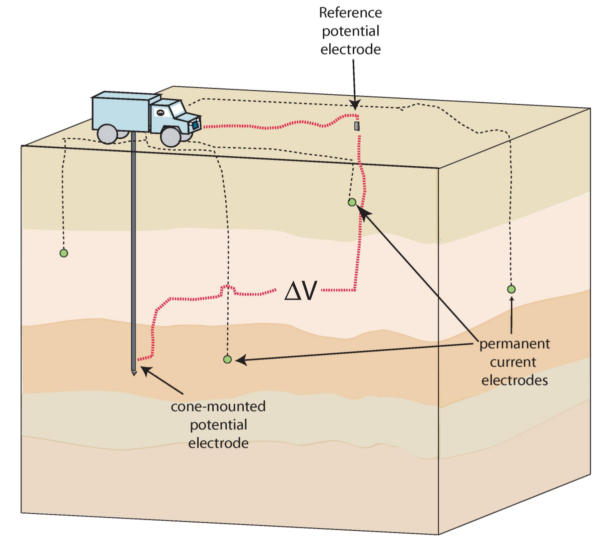
Joint project with R. Knight and A. Pidlovski, Stanford Environmental Geophysics Group



Application: Impedance Tomography



Application: Impedance Tomography



Application: Impedance Tomography

The mathematical problem

The constraint (PDE)

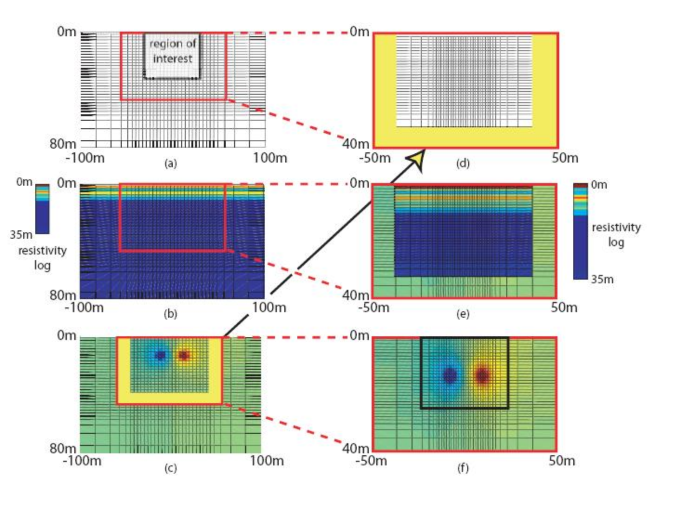
$$c(y,u) = \nabla \times \ \mu^{-1} \nabla \times \ y - i\omega\sigma y = i\omega s_j \quad j = 1 \ldots k$$

(with some BC)

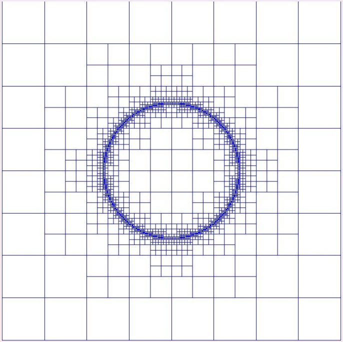
The Objective function

$$\min \quad \frac{1}{2} \underbrace{\|Q(y - y^{\mathrm{obs}})\|^2}_{\text{misfit}} + \underbrace{\alpha}_{\text{regpar}} \quad \overbrace{R(u)}^{\text{regularization}}$$

Discretization - I



Discretization - II



Discretization

use $128 \times 128 \times 64$ cells
$\#$ of states $= k \times \#$ of controls

In practical experiments $k \approx 10 - 1000$

The discrete mathematical problem

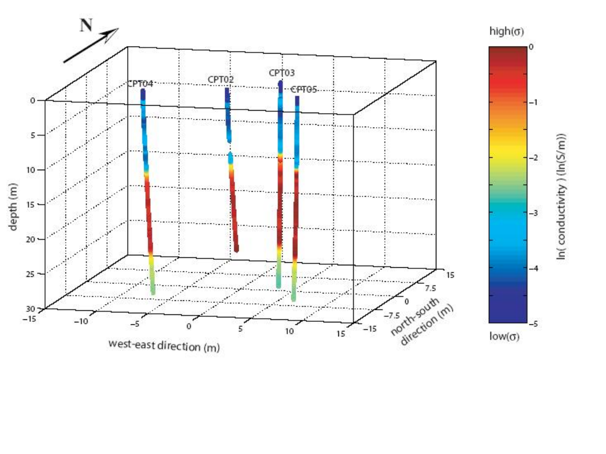
The constraint (PDE)

$$c_h(y_h, u_h) = A(u_h)y_h - q_h = D^T S(u_h) D y_h - q_h = 0$$

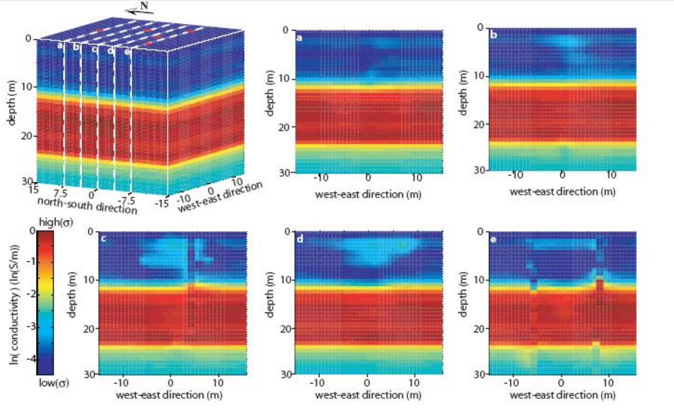
The Objective function

$$\min \quad \frac{1}{2} \underbrace{\|Q(y_h - y^{\mathrm{obs}})\|^2}_{\text{misfit}} + \underbrace{\alpha}_{\text{regpar}} \quad \overbrace{R(u_h)}^{\text{regularization}}$$

The Data - 63 sources



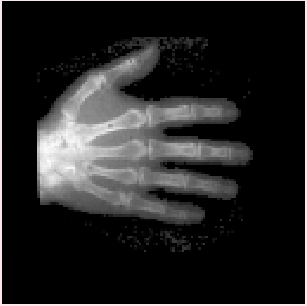
The Inversion



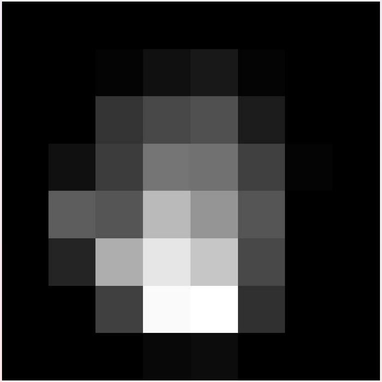
Application - Image Registration

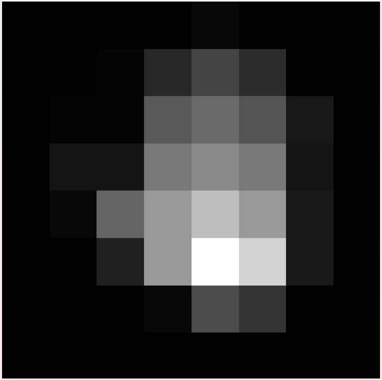
Joint work with S. Heldmann and J. Modesitzki, Lübeck, Germany

$$\begin{array}{ll} \min & \frac{1}{2}\|y(T) - R\|^2 + \frac{1}{2}\alpha S(u) \\ \text{s.t} & y_t + u^\top \nabla y = 0 \quad y(0) = y_0 \end{array}$$

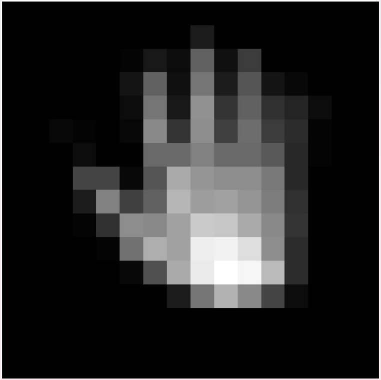


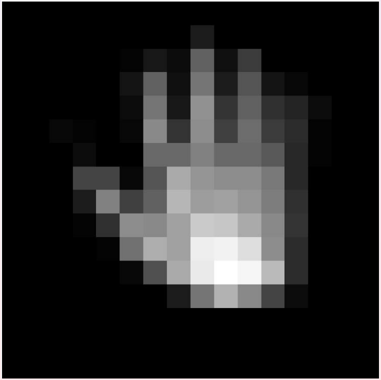
Example - ML



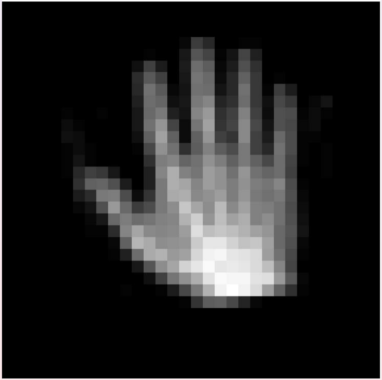
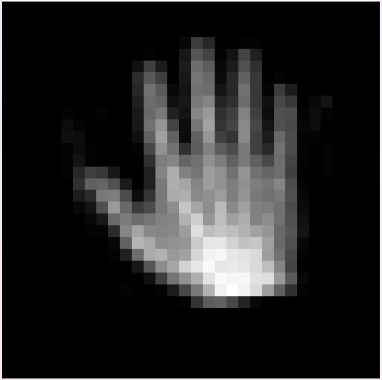


Example - ML

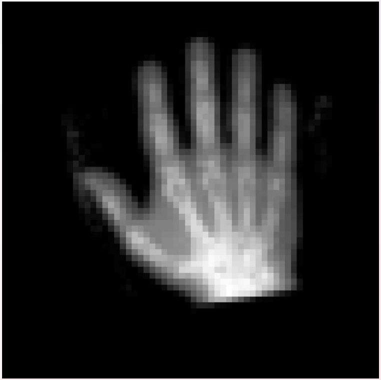


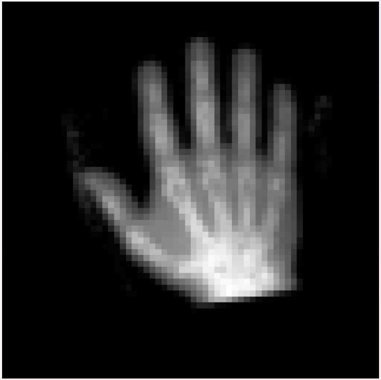


Example - ML

Example - ML

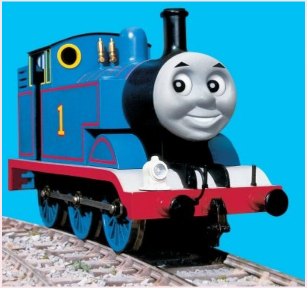




Example - ML

Model Problems

Sometimes, you can learn a lot from small things Thomas the engine



Goal

- PDE optimization problems are difficult to implement
- Suggest some *simple* model problems we can experiment with
- Develop optimization algorithms, preconditioners, grounded to reality
- Will not cover all PDE-optimization problems but not all PDE's are Poisson equation either
- Much of the development in PDE's was motivated by the 5 point stencil!

The problems/implementation

Parameter identification problems

- Assume smooth enough problems (discretize optimize not a problem)
- Consider elliptic, parabolic and hyperbolic problems
- Use regular grids and finite difference/volume for simplicity
- Code in matlab
- Modular, BYOPC (bring your own preconditioner)

The problems

The PDE's

- Elliptic

$$\nabla \cdot \exp^u \nabla y - q = 0; \quad \mathbf{n} \cdot y = 0$$

- Parabolic

$$y_t - \nabla \cdot \exp^u \nabla y = 0; \quad \mathbf{n} \cdot y = 0; \quad y(x,0) = y_0$$

- Hyperbolic

$$y_t - \vec{u}^\top \nabla y = 0; \quad \mathbf{n} \cdot y = 0; \quad y(x,0) = y_0$$

The code

Download:

`http://www.mathcs.emory.edu/ haber/code.html`

Very simple to get started (matlab demo ...)

Takes some time to run, elliptic problem on n^3 grid has $6n^3 + n^3 + 6n^3$ variables

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work

Outline/Summary

- Introduction
- Difficulties and PDE aspects
- The optimization framework
- Solving the KKT system
- Optimization algorithms
- Examples
- Summary and future work