Faster gradient descent and artificial time integration

Uri Ascher

Department of Computer Science
University of British Columbia

ascher@cs.ubc.ca
www.cs.ubc.ca/∼ascher

with **Kees van den Doel, Hui Huang, Benar Svaiter**

Outline

- Motivation
  - Box constrained convex quadratic optimization
  - Unconstrained gradient descent
  - Artificial time
  - Finite time regularization

Box constrained convex quadratic optimization

[Friedlander & van den Berg, '08; Figueiredo, Nowak & Wright, '07; Dai & Fletcher '05]:

To find sparse solutions for compressed sensing etc. consider

$$\min_{\mathbf{x}} \|\mathcal{A}\mathbf{x} - \mathbf{y}\|^2, \quad s.t.\ \|\mathbf{x}\|_1 \leq tol,$$

i.e. minimize convex quadratic subject to box constraints, and use a **fast gradient descent** method.

Box constrained convex quadratic optimization

[Friedlander & van den Berg, '08; Figueiredo, Nowak & Wright, '07; Dai & Fletcher '05]:

To find sparse solutions for compressed sensing etc. consider

$$\min_{\mathbf{x}} \|\mathcal{A}\mathbf{x} - \mathbf{y}\|^2, \quad s.t. \ \|\mathbf{x}\|_1 \leq tol,$$

i.e. minimize convex quadratic subject to box constraints, and use a **fast gradient descent** method.

?? But is there a fast gradient descent method **??**

Unconstrained gradient descent

For the basic problem

$$A\mathbf{x} = \mathbf{b},$$

A symmetric positive definite (SPD),

well known that steepest descent (SD) is very slow, conjugate gradient (CG) much faster.

Gradient descent

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k, \quad \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k.$$

Steepest and lagged steepest descent

Gradient descent

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k, \quad \mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k.$$

Steepest descent

$$\alpha_k^{SD} = \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{r}_k, A\mathbf{r}_k)}.$$

Claim: much faster to use non-monotonically convergent LSD

Lagged steepest descent [Barzilai & Borwein, '88]

$$\alpha_k^{LSD} = \frac{(\mathbf{r}_{k-1}, \mathbf{r}_{k-1})}{(\mathbf{r}_{k-1}, A\mathbf{r}_{k-1})}.$$

Minimization and artificial time

Unconstrained minimization:

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{necessary } \boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{0}.$$

Minimization and artificial time

Unconstrained minimization:

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{necessary } \boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{0}.$$

Convex quadratic case:

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x}, A\mathbf{x}) - (\mathbf{b}, \mathbf{x}), \quad \boldsymbol{\nabla} f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = -\mathbf{r}.$$

Steepest descent: gradient descent with exact line search wrto f.

Minimization and artificial time

Unconstrained minimization:

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{necessary } \boldsymbol{\nabla} f(\mathbf{x}) = \mathbf{0}.$$

Convex quadratic case:

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x}, A\mathbf{x}) - (\mathbf{b}, \mathbf{x}), \quad \boldsymbol{\nabla} f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = -\mathbf{r}.$$

Steepest descent: gradient descent with exact line search wrto f.

Necessary condition may be considered as steady state for ODE:

$$\frac{d\mathbf{x}}{dt} = -\boldsymbol{\nabla} f(\mathbf{x}).$$

Artificial time DE

$$\frac{d\mathbf{x}}{dt} = -\boldsymbol{\nabla} f(\mathbf{x}).$$

Forward Euler discretization

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \boldsymbol{\nabla} f(\mathbf{x}_k).$$

Equivalent to gradient descent.

How to choose step size? What about absolute stability?

Numerical example

$$u_t = \Delta u + 1 \quad 0 < x, y < 1, \; t \geq 0,$$

+ homogeneous BC.

Discretize in space on a uniform mesh and call unknowns $\mathbf{x} \in \mathbb{R}^m$. Apply forward Euler in time, stopping when $\|\mathbf{r}_k\| \leq 10^{-6}\|\mathbf{r}_0\|$.

m	SD	CG
49	167	9
225	702	24
961	2,859	50
3969	11,517	100

Step (iteration) counts for the heat $\Rightarrow$ Poisson equation

Numerical example

$$u_t = \Delta u + 1 \quad 0 < x, y < 1, \ t \geq 0,$$

+ homogeneous BC.

Discretize in space on a uniform mesh and call unknowns $\mathbf{x} \in \mathbb{R}^m$. Apply forward Euler in time, stopping when $\|\mathbf{r}_k\| \leq 10^{-6}\|\mathbf{r}_0\|$.

m	SD	LSD	CG
49	167	40	9
225	702	72	24
961	2,859	240	50
3969	11,517	356	100

Step (iteration) counts for the heat $\Rightarrow$ Poisson equation

Smoothing and finite time regularization

Consider linear problem $J\mathbf{x} = \mathbf{y}$, J constant and ill-conditioned.

Artificial time ODE

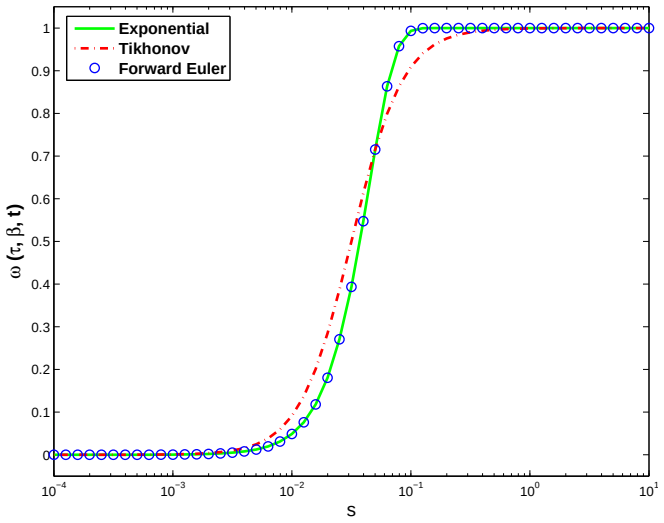
$$\frac{d\mathbf{x}}{dt} = -J^T(J\mathbf{x} - \mathbf{y}) \equiv \mathbf{b} - A\mathbf{x}.$$

Using SVD, regularization replaces singular values s_i^{-1} by $\omega(s_i^2)s_i^{-1}$, where

$$\omega(s) = 1 - e^{-ts}.$$

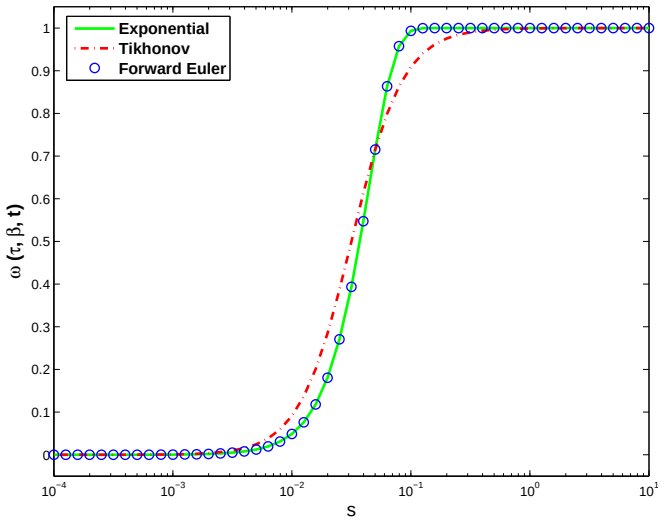
So, the effect of small singular values gets dampened while large ones remain almost intact for appropriate, **finite time**.

Filter functions



Exponential filter $\omega(s) = 1 - e^{-ts}$ and Tikhonov filter $\omega(s) = \frac{s}{s+\beta}$ for $t\beta = 1/2$.

Filter functions



Exponential filter $\omega(s) = 1 - e^{-ts}$ and Tikhonov filter $\omega(s) = \frac{s}{s+\beta}$ for $t\beta = 1/2$.

Question: Does this regularization effect still hold when ODE is integrated with large steps? Is there an advantage in doing so?

Outline

- Motivation
- Linear positive definite systems
- Regularization effects of gradient descent / artificial time in applications
- Conclusions

Outline

- Linear positive definite systems
  - Step size choices for gradient descent / forward Euler
  - Slowness of greedy algorithms
  - Faster step size selection

$$A\mathbf{x} = \mathbf{b}$$

Step size choices

Define **energy norm** for any SPD matrix P

$$(\mathbf{v}, \mathbf{w})_P = \mathbf{v}^T P \mathbf{w}, \quad \|\mathbf{v}\|_P^2 = (\mathbf{v}, \mathbf{v})_P.$$

Then minimizing $f(\mathbf{x})$ is equivalent to minimizing $\|\mathbf{r}\|_{A^{-1}}^2$.

Step size choices

Define **energy norm** for any SPD matrix P

$$(\mathbf{v}, \mathbf{w})_P = \mathbf{v}^T P \mathbf{w}, \quad \|\mathbf{v}\|_P^2 = (\mathbf{v}, \mathbf{v})_P.$$

Then minimizing $f(\mathbf{x})$ is equivalent to minimizing $\|\mathbf{r}\|_{A^{-1}}^2$.

Another **greedy algorithm**: exact line search wrto $\|\mathbf{r}\|^2$

$$\alpha_k^{OM} = \frac{(\mathbf{r}_k, A\mathbf{r}_k)}{(A\mathbf{r}_k, A\mathbf{r}_k)}.$$

Orthomin (OM) and SD combinations

$$\alpha_k^{SD} = \frac{\|\mathbf{r}_k\|^2}{\|\mathbf{r}_k\|_A^2},$$

$$\alpha_k^{OM} = \frac{\|\mathbf{r}_k\|_A^2}{\|A\mathbf{r}_k\|^2} = \frac{\|\mathbf{r}_k\|_A^2}{\|\mathbf{r}_k\|_{A^2}^2}.$$

Combinations:

$$\alpha_k^{HM} = \frac{2}{1/\alpha_k^{SD} + 1/\alpha_k^{OM}},$$

$$\alpha_k^{SD/OM} = \begin{cases} \alpha_k^{SD} & \text{k even} \\ \alpha_k^{OM} & \text{k odd} \end{cases},$$

$$\alpha_k^{RSDOM} = \text{random combination of } \alpha_k^{SD} \text{ and } \alpha_k^{OM}.$$

Half lagged steepest descent (HLSD)

Like LSD, a two-step method: use same α_{2j}^{SD} for next step: $\alpha_{2j+1}^{HLSD} = \alpha_{2j}^{HLSD}$.

[Friedlander, Martinez, Molina & Raydan, '99; Raydan & Svaiter, '02]

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	LSD
49	167	40
225	702	72
961	2,859	240
3969	11,517	356

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	OM	LSD
49	167	169	40
225	702	696	72
961	2,859	2,811	240
3969	11,517	11,279	356

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	OM	HM	LSD
49	167	169	169	40
225	702	696	698	72
961	2,859	2,811	2,819	240
3969	11,517	11,279	11,299	356

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	OM	HM	SD/OR	LSD
49	167	169	169	46	40
225	702	696	698	88	72
961	2,859	2,811	2,819	276	240
3969	11,517	11,279	11,299	878	356

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	OM	HM	SD/OM	RSDOM	LSD
49	167	169	169	46	57	40
225	702	696	698	88	126	72
961	2,859	2,811	2,819	276	311	240
3969	11,517	11,279	11,299	878	682	356

Numerical example: heat $\Rightarrow$ Poisson equation

Iteration counts for more step-size selections, same model problem

m	SD	OM	HM	SD/OM	RSDOM	LSD	HLSD
49	167	169	169	46	57	40	59
225	702	696	698	88	126	72	67
961	2,859	2,811	2,819	276	311	240	142
3969	11,517	11,279	11,299	878	682	356	590

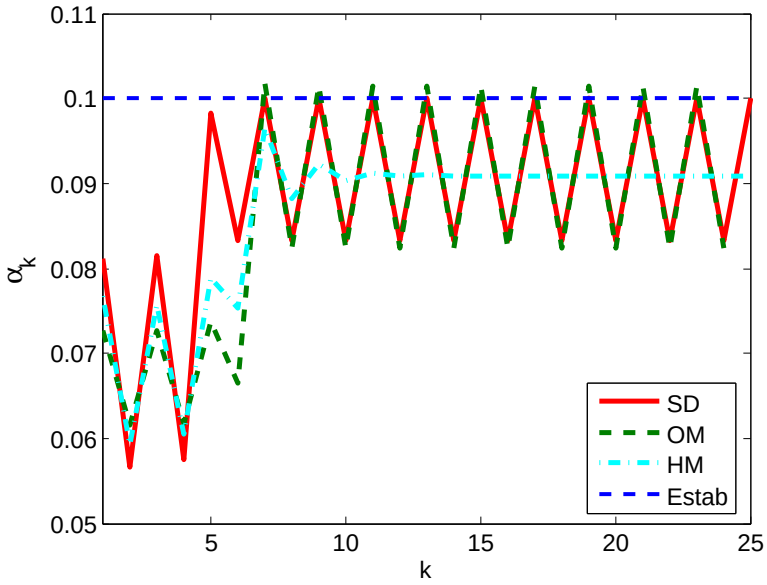
It's not true that:

- A fast method must be multi-step
- A fast method must be non-monotonic
- LSD works because it's a secant approximation

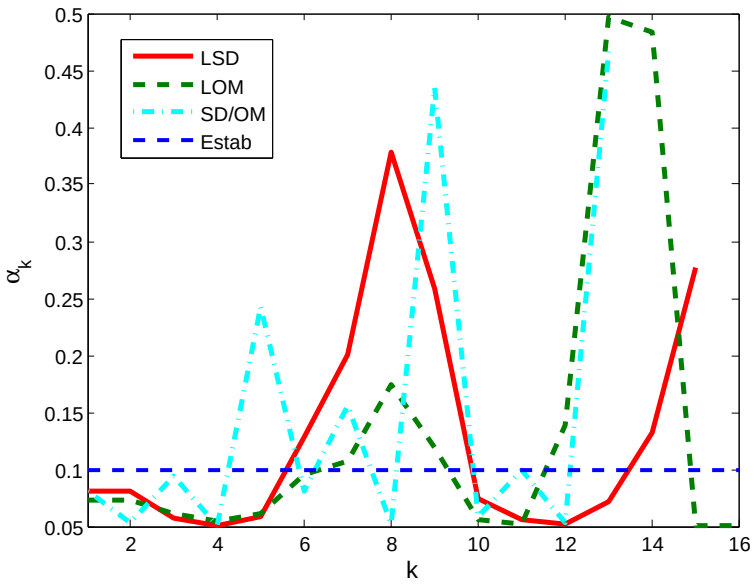
m	SD	OM	HM	SD/OM	RSDOM	LSD	HLSD
49	167	169	169	46	57	40	59
225	702	696	698	88	126	72	67
961	2,859	2,811	2,819	276	311	240	142
3969	11,517	11,279	11,299	878	682	356	590

Step size dynamics

Step size sequences for $A = \mathrm{diag}(20, 10, 2, 1)$



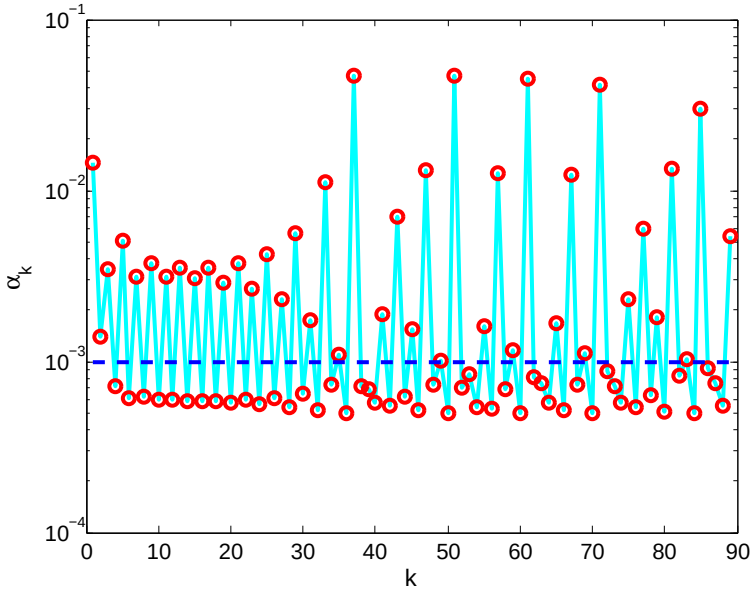
(a) Greedy step sizes



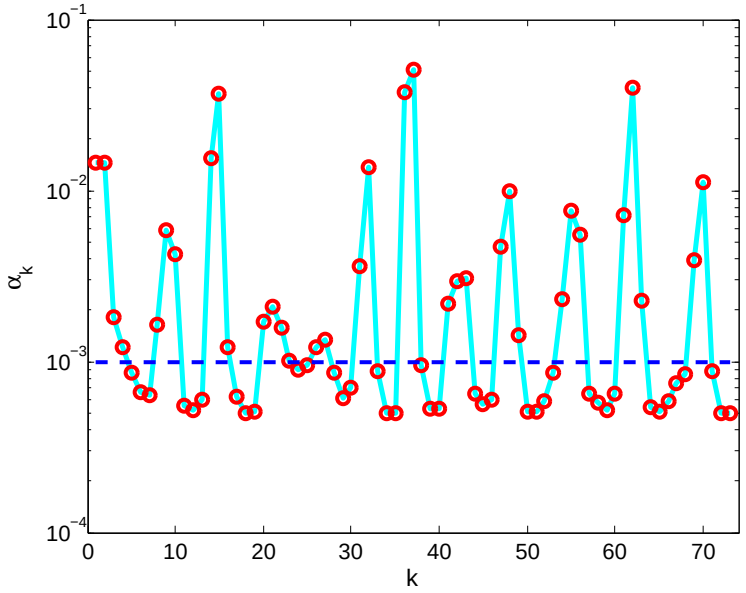
(b) Wilder step sizes

Step size dynamics

Step size sequences for heat $\Rightarrow$ Poisson eqn example, $m = 225$. Note large violation of forward Euler stability limit.



(c) Step sizes SD/OR



(d) Step sizes LSD

Slowness of the greedy algorithms

Denote eigenvalues of A by $\lambda_1 > \lambda_2 > \cdots > \lambda_m > 0$.

Case of constant (uniform) step size

Theorem Barring special initial conditions, the fastest reduction in residual norm towards steady state using forward Euler with a constant step size is obtained using the step size

$$\alpha^* = \frac{2}{\lambda_1 + \lambda_m}.$$

For this step size the number of steps required to reduce the residual error by a constant amount is proportional to the condition number $\kappa = cond(A) = \frac{\lambda_1}{\lambda_m}$.

Residuals' contraction

Indeed, write iteration in terms of residual:

$$\mathbf{r}_{k+1} = (I - \alpha_k A)\mathbf{r}_k.$$

Assume wlog a diagonal A. Then

$$r_i^{(k+1)} = (1 - \alpha_k \lambda_i) r_i^{(k)}.$$

So

$$r_m^{(k+1)} = (1 - \alpha^* \lambda_m) r_m^{(k)} = \left(\frac{\lambda_1 - \lambda_m}{\lambda_1 + \lambda_m} \right) r_m^{(k)} = \left(\frac{\kappa - 1}{\kappa + 1} \right) r_m^{(k)}.$$

Slowness of the greedy algorithms cont.

Case of variable step size

Theorem Barring special initial conditions:

1. For SD there are constants satisfying $\frac{2}{\beta_0^{-1}+\beta_1^{-1}} = \alpha^*$ such that as $j \to \infty$, $\alpha_{2j} \to \beta_0$; $\alpha_{2j+1} \to \beta_1$.

2. Similarly for OM.

[Akaike, '59]

Theorem cont.

3. For HM, assume further that the sequence of residual vectors tends to satisfy

$$\mathbf{r}_{k+1} = \gamma D_k \mathbf{r}_k, \quad k \to \infty,$$

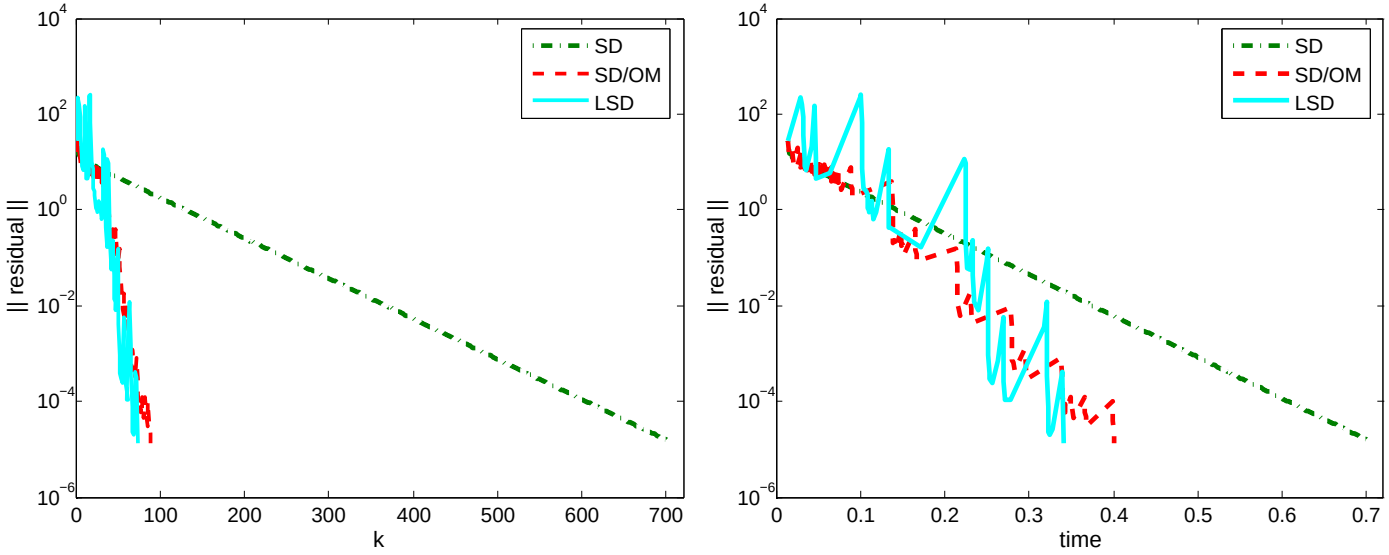
   γ scalar, D_k diagonal with elements ± 1.

   Then the step sizes tend to α^*. Also γ and normalized residual (up to signs) are determined.

4. For all three variants the number of steps necessary to decrease the error by a constant amount grows linearly in the condition number κ.

Faster step size selection

Both one-step and multi-step strategies can yield faster convergence.
Payment: **non-monotonic convergence**.



(e) $||\mathbf{r}||$ as a function of step #

(f) $||\mathbf{r}||$ as a function of time

Residual norms and step size sequences.

Half lagged steepest descent (HLSD)

- Behaves essentially like LSD
- Write as double step

$$\mathbf{r}_{k+2} = (I - \alpha_k^{SD} A)^2 \mathbf{r}_k = [I - 2\alpha_k^{SD} A + (\alpha_k^{SD} A)^2]\mathbf{r}_k,$$

  which is a 2-stage 1st-order Runge-Kutta with step size $2\alpha_k^{SD}$.

- Converges monotonically in (useless) norm $\|\mathbf{e}_k\|_{A^{-1}}$.
  [Raydan & Svaiter, '02]

Outline

- Motivation
- Linear positive definite systems
- Regularization effects of gradient descent / artificial time in applications
- Conclusions

Outline

- Regularization effects of gradient descent / artificial time in applications
  - Image denoising and deblurring in 2D
    Hui Huang
  - Shape optimization in 3D for EIT and DC resistivity
    Kees van den Doel

Image denoising in 2D

Find $w(x, y)$ that cleans given image $b(x, y)$.

(g) True image

(h) Given b = with 20% noise

Cameraman 256×256 with noise added.

Image deblurring in 2D

Find $w(x, y)$ that sharpens given image $b(x, y)$.

(i) True image

(j) MOTION blur

Boat 256×256 blurred.

Huber function regularization

Tikhonov: $\min \ \frac{1}{2}\|Jw - b\|^2 + \beta R(w)$.

$$R(w) = \int_\Omega \rho(|\boldsymbol{\nabla} w|)$$

Use the Huber function

$$\rho(s) \quad = \quad \begin{cases} s, & s \geq \gamma, \\ s^2/(2\gamma) + \gamma/2, & s < \gamma \end{cases} \quad \Rightarrow$$

$$R_w(w) \quad \leftarrow \quad -\nabla \cdot \left(\min\{\frac{1}{\gamma}, \frac{1}{|\boldsymbol{\nabla} w|}\} \, \boldsymbol{\nabla} w \right)$$

Selecting the switching parameter

Choose γ wisely:

1. Letting $\gamma \to \infty$ obtain least squares (LS)

2. Determine adaptively, close to **total variation** (TV)

$$\gamma = \frac{h}{|\Omega|} \int_\Omega |\boldsymbol{\nabla} w|.$$

"Our Huber" [Ascher, Haber & Huang, '06]

Gradient descent method

Necessary condition for Tikhonov minimum

$$G(w) \equiv J^T(Jw - b) + \beta R_w = \mathbf{0}.$$

Gradient descent method

Necessary condition for Tikhonov minimum

$$G(w) \equiv J^T(Jw - b) + \beta R_w = \mathbf{0}.$$

Iteration

$$w_{k+1} = w_k - \alpha_k \left(J^T(Jw_k - b) + \beta \hat{R}(w_k) w_k \right), \quad k = 0, 1, 2, \ldots,$$

$$R_w(w) = \hat{R}(w) \cdot w.$$

Step size selection

Compare the two step size choices obtained by freezing nonlinearities:

$$\text{SD:} \qquad \alpha_k = \frac{(G(w_k))^T G(w_k)}{(G(w_k))^T (J^T J + \beta \hat{R}(w_k)) G(w_k)} \ ,$$

$$\text{LSD:} \qquad \alpha_k = \frac{(G(w_{k-1}))^T G(w_{k-1})}{(G(w_{k-1}))^T (J^T J + \beta \hat{R}(w_{k-1})) G(w_{k-1})} \ .$$

Denoising by anisotropic diffusion

Here $J =$ identity. Apply forward Euler starting from data

$$\begin{aligned} w_0 &= b, \\ w_{k+1} &= w_k - \alpha_k R_w(w_k), \quad k = 0, 1, 2, \ldots. \end{aligned}$$

Or gradient descent for minimizing $R(w)$, i.e. set $\beta \to \infty$ in rescaled formulation and start from data.

Do not integrate to steady state

Step size choices

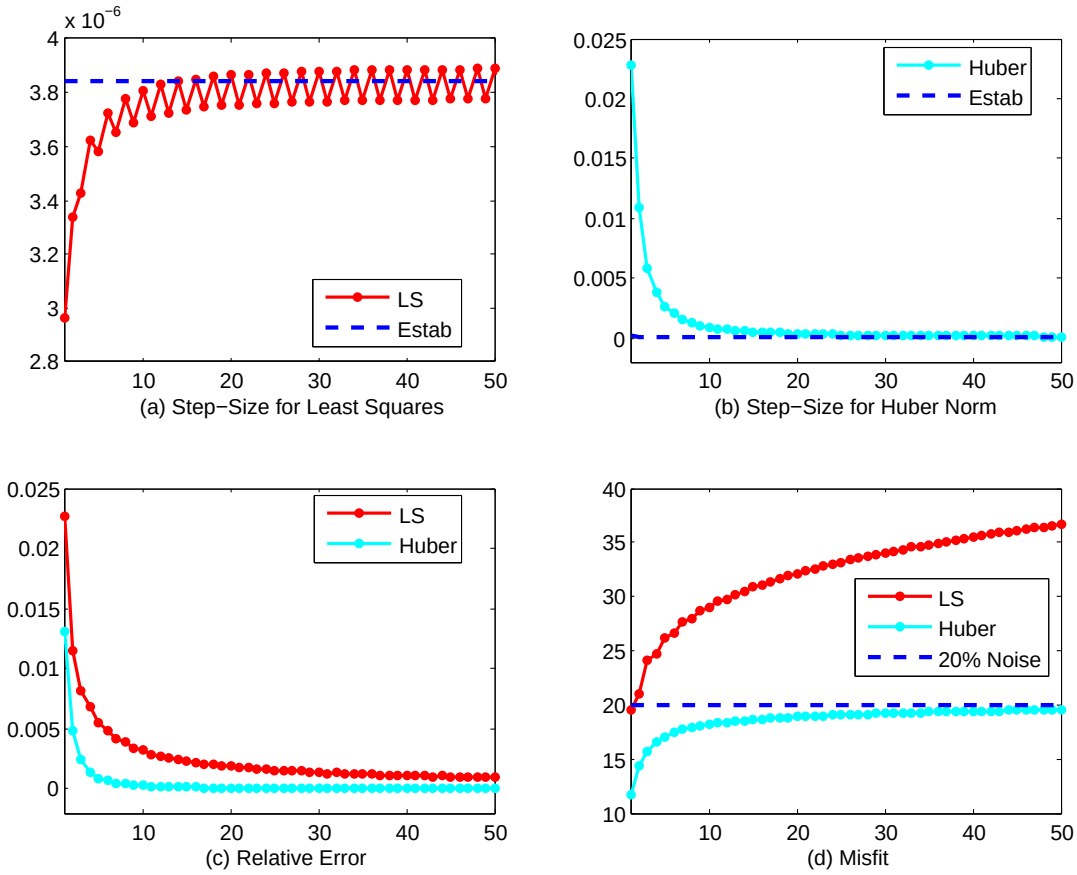
$$\alpha_k^{SD} = \frac{(R_w(w_k))^T (R_w(w_k))}{(R_w(w_k))^T \hat{R}(w_k)(R_w(w_k))} \quad ,$$

$$\alpha_k^{LSD} = \frac{(R_w(w_{k-1}))^T (R_w(w_{k-1}))}{(R_w(w_{k-1}))^T \hat{R}(w_{k-1})(R_w(w_{k-1}))} \quad ;$$

$$R_w(w) = \hat{R}(w) \cdot w$$

The purpose is efficient anisotropic smoothing.

Discrete SD dynamics for `Cameraman` with 20% white noise added.



(k) SD, $\mathrm{Misfit}_{21} = 18.94$

(l) LSD, $\mathrm{Misfit}_{14} = 20.44$

(m) SD, $\mathrm{Misfit}_{382} = 20.96$

(n) LSD, $\mathrm{Misfit}_{117} = 21.04$

Using diffusion algorithm with adaptive γ.

Image deblurring in 2D

[Huang & Ascher, '08]

(o) MOTION blur

(p) 33 LSD or 113 SD iters

TYPE = 'MOTION', LEN = 15, THETA = 30, $\eta = 1$, $\beta = 10^{-4}$.

Image deblurring in 2D

Roughly 1.3 sec per iter for 256×256 images.

(q) True image

(r) DISK blur

(s) 99 SD or 26 LSD iters

TYPE = 'DISK', RADIUS = 5, $\eta = 1$, $\beta = 10^{-4}$.

Other related denoising and deblurring methods

- For denoising can use rough tolerance results to switch to another method.

- For deblurring in presence of much noise can try a splitting approach.

In both cases not too many SD iterations are required, and LSD advantage is limited.

Shape optimization in 3D for EIT and DC resistivity

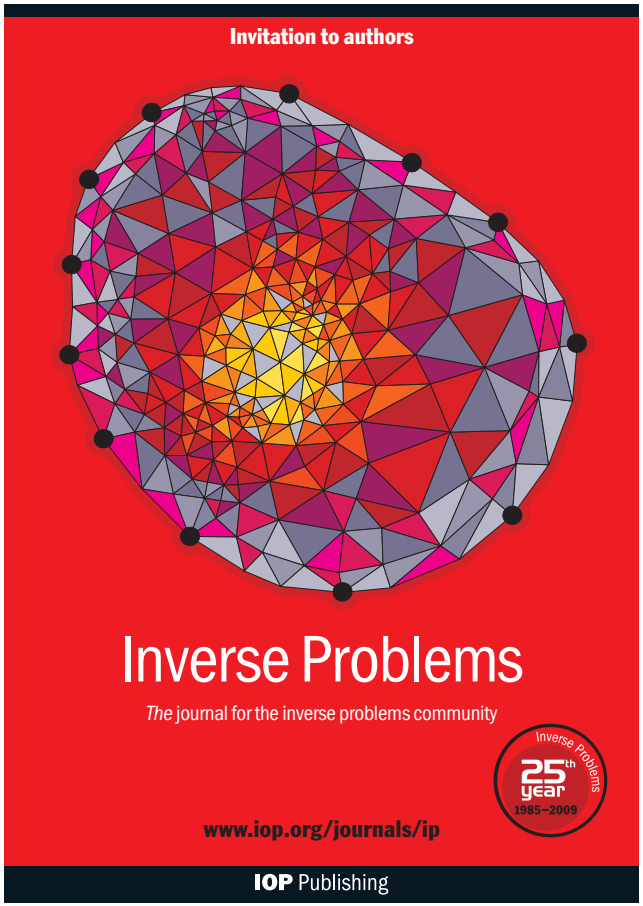
Forward problem:

$$\begin{aligned} \nabla \cdot (\sigma \boldsymbol{\nabla} u_j) &= q_j, \quad j = 1, \ldots, s, \\ \frac{\partial u_j}{\partial \nu}|_{\partial\Omega} &= 0. \end{aligned}$$

Domain $\Omega = [0,1]^3$.

Predict data by measuring field at specified locations:
$F(w) = (F_1, \ldots, F_s)^T$, $w = \log \sigma$.

A 2D picture


Invitation to authors
Inverse Problems
The journal for the inverse problems community
Inverse Problems
25th year
1985–2009
www.iop.org/journals/ip
IOP Publishing

Shape optimization

[van den Doel & Ascher, '07]

Assume $w(x, y, z)$ can take only one of two values at each (x, y, z).

Inverse problem: Recover w from measured data b.

Shape optimization

[van den Doel & Ascher, '07]

Assume $w(x, y, z)$ can take only one of two values at each (x, y, z).

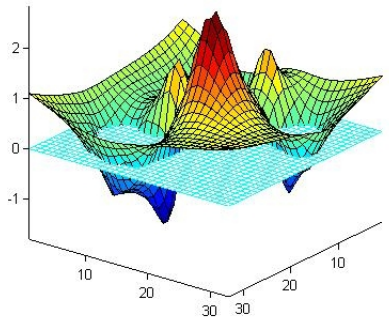
Inverse problem: Recover w from measured data b.

Thus, **shape optimization**.

A level set approach

$$w = \chi(\psi), \quad e.g.$$
$$\chi(s) = \frac{w_I - w_{II}}{2}\tanh(s/h) + \frac{w_I + w_{II}}{2}.$$

Thus, sharpening happens at each iteration l across the 0- level set of $\psi_l(x, y, z)$.



Dynamic regularization

Damped **Gauss-Newton** for

$$\min_{\psi} \frac{1}{2} \sum_{j=1}^{s} \|F_j(w(\psi)) - b_j\|^2$$

yields iteration

$$\left(\sum_{j=1}^{s} \hat{J}_j^T \hat{J}_j \right) \delta\psi = -\tau \sum_{j=1}^{s} \hat{J}_j^T (F_j(w(\psi)) - b_j)$$

where

$$\hat{J} = \frac{\partial F}{\partial \psi} = \frac{\partial F}{\partial w} \frac{\partial w}{\partial \psi}.$$

Dynamic regularization

Outer iteration: for $l = 0, 1, \ldots$ do

$$\left(\sum_{j=1}^{s} \hat{J}_j^T \hat{J}_j \right) \delta\psi = -\sum_{j=1}^{s} \hat{J}_j^T \left(F_j(w(\psi_l)) - b_j \right),$$

$$\psi_{l+1} = \psi_l + \tau \delta\psi$$

Inner iteration (for a singular system): a fixed number of preconditioned gradient descent (**PSD** or **PLSD**) or conjugate gradient **PCG** iterations.

Use discrete Laplacian as preconditioner for $\delta\psi$.

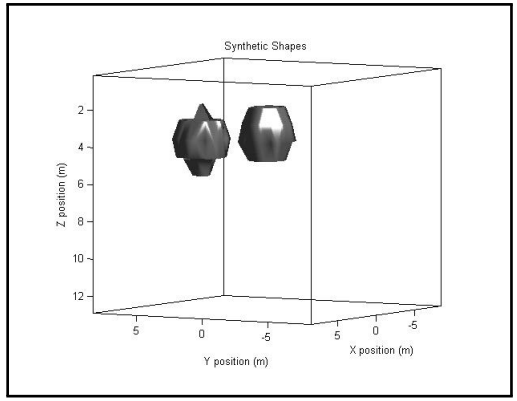
Experiments

Setups and RESINVM3D from [Pidilsecky, Haber & Knight, '07]

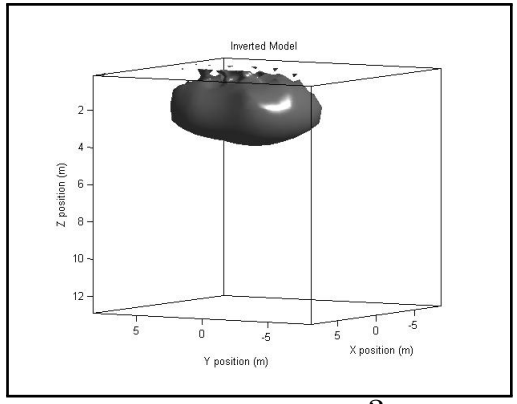
$$\nabla \cdot (e^w \boldsymbol{\nabla} u_j) = q_j, \quad j = 1, \ldots, s,$$
$$\frac{\partial u_j}{\partial \nu}|_{\partial\Omega} = 0,$$

discretized on a staggered grid

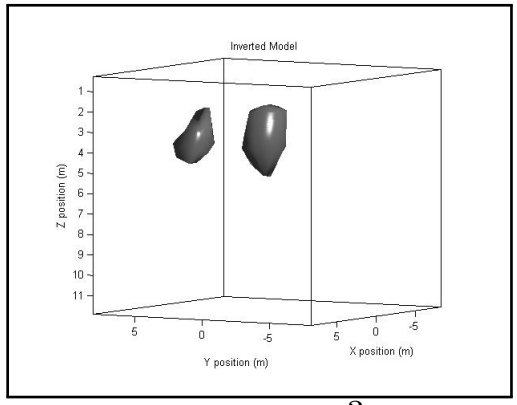
1. Surface electrodes, $w_I = -2.3, w_{II} = -5.3$, $s = 41$, 3% noise
2. Subsurface electrodes, 3% noise
3. Subsurface electrodes and receivers, 1% noise



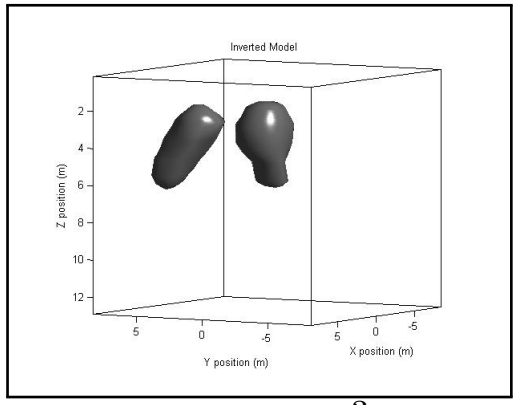
(t) Synthetic model.



(u) L_2-method, 32^3 grid.

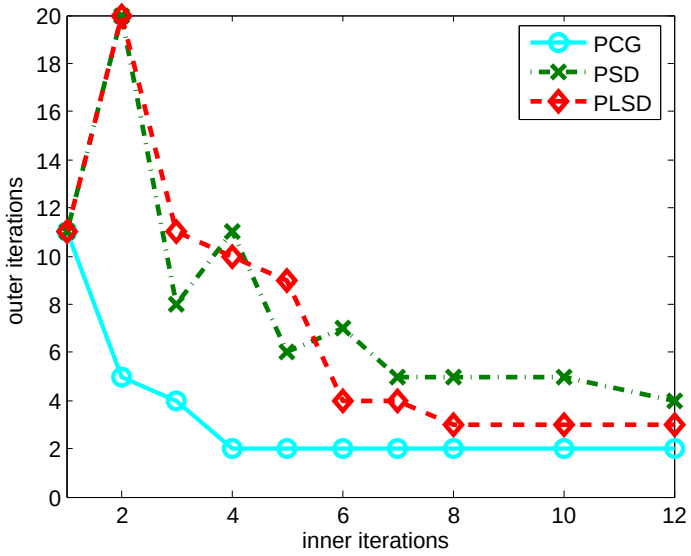


(v) Level set, 16^3 grid.

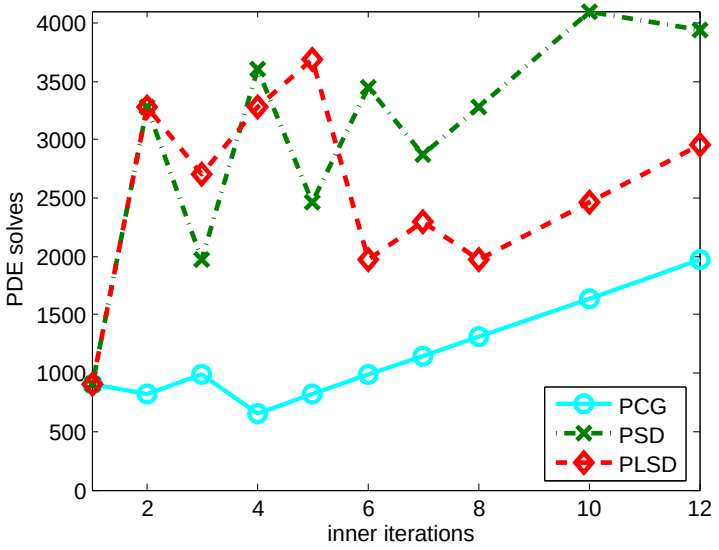


(w) Level set, 32^3 grid.

PSD, PCG and PLSD



(x) Outer iterations required



(y) Work units

Inner iteration counts and number of PDE solves = $s*2*iters_{in}*iters_{out}$.
For $iters_{in} = 1$ all three methods coincide.

Outline

- Motivation
- Linear positive definite systems
- Regularization effects of gradient descent / artificial time
- Conclusions

Conclusions

- Step size selection for **gradient descent**, or integrating to **steady state** using **forward Euler**, has been investigated.

- For unconstrained quadratic both one-step and two-step **gradient descent** methods can converge significantly faster than **steepest descent** (although not as fast as **CG**).

- Greedy algorithms SD, OM and HM are shown to be inherently slow.

- Occasionally some advantage can be gained by using LSD or some other faster method rather than SD as a smoother or regularizer.

Conclusions

- LSD or HLSD yield worthwhile advantage over SD when more than, say, 15 SD steps are required for an essentially quadratic minimization.

- The practical gains from LSD or HLSD do not appear to be huge, although they may be significant.

- There is need for a better theory for the **faster gradient descent** variants.

Happy birthday, Ernst