

EE364 Review Session 7

Outline:

- derivatives and chain rule (Appendix A.4)
- unconstrained minimization
- approximate TV de-noising

Derivative and gradient

When f is real-valued (*i.e.*, $f : \mathbf{R}^n \rightarrow \mathbf{R}$), the derivative $Df(x)$ is a $1 \times n$ matrix, *i.e.*, it is a *row* vector

- its transpose is called the *gradient* of the function:

$$\nabla f(x) = Df(x)^T$$

which is a (column) vector, *i.e.*, in \mathbf{R}^n

- its components are the partial derivatives of f :

$$\nabla f(x)_i = \frac{\partial f(x)}{\partial x_i}, \quad i = 1, \dots, n$$

- the first-order approximation of f at a point x can be expressed as (the affine function of z)

$$f(x) + \nabla f(x)^T(z - x)$$

example: Find the gradient of $g : \mathbf{R}^m \rightarrow \mathbf{R}$

$$g(y) = \log \sum_{i=1}^m \exp(y_i)$$

solution.

$$\nabla g(y) = \frac{1}{\sum_{i=1}^m \exp y_i} \begin{bmatrix} \exp y_1 \\ \vdots \\ \exp y_m \end{bmatrix}$$

Chain rule

Suppose $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$ and $g : \mathbf{R}^m \rightarrow \mathbf{R}^p$ are differentiable. Define $h : \mathbf{R}^n \rightarrow \mathbf{R}^p$ by $h(x) = g(f(x))$. Then

$$Dh(x) = Dg(f(x))Df(x)$$

- Composition with an affine function:

Suppose $g : \mathbf{R}^m \rightarrow \mathbf{R}^p$ is differentiable, $A \in \mathbf{R}^{m \times n}$, and $b \in \mathbf{R}^m$. Define $h : \mathbf{R}^n \rightarrow \mathbf{R}^p$ as $h(x) = g(Ax + b)$.

The derivative of h is $Dh(x) = Dg(Ax + b)A$

When g is real-valued (*i.e.*, $p = 1$)

$$\nabla h(x) = A^T \nabla g(Ax + b)$$

example A.2: Find the gradient of $h : \mathbf{R}^n \rightarrow \mathbf{R}$

$$h(x) = \log \sum_{i=1}^m \exp(a_i^T x + b_i)$$

where $a_1, \dots, a_m \in \mathbf{R}^n$, and $b_1, \dots, b_m \in \mathbf{R}$

Hint: h is the composition of the affine function $Ax + b$, where $A \in \mathbf{R}^{m \times n}$ has rows a_1^T, \dots, a_m^T , and the function $g(y) = \log(\sum_{i=1}^m \exp y_i)$

solution.

$$\nabla g(y) = \frac{1}{\sum_{i=1}^m \exp y_i} \begin{bmatrix} \exp y_1 \\ \vdots \\ \exp y_m \end{bmatrix}$$

then by the composition formula we have

$$\nabla h(x) = \frac{1}{\mathbf{1}^T z} A^T z$$

where $z_i = \exp(a_i^T x + b_i)$, $i = 1, \dots, m$

Second derivative

When f is real-valued (*i.e.*, $f : \mathbf{R}^n \rightarrow \mathbf{R}$), the second derivative or Hessian matrix $\nabla^2 f(x)$ is a $n \times n$ matrix, with components

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}, \quad i = 1, \dots, n, \quad j = 1, \dots, n$$

- the second-order approximation of f , at or near x , is the quadratic function of z defined by

$$\hat{f}(z) = f(x) + \nabla f(x)^T (z - x) + (1/2)(z - x)^T \nabla^2 f(x) (z - x)$$

Chain rule for second derivative

- Composition with scalar function

Suppose $f : \mathbf{R}^n \rightarrow \mathbf{R}$, $g : \mathbf{R} \rightarrow \mathbf{R}$, and $h(x) = g(f(x))$

The second derivative of h is

$$\nabla^2 h(x) = g'(f(x))\nabla^2 f(x) + g''(f(x))\nabla f(x)\nabla f(x)^T$$

- Composition with affine function

Suppose $g : \mathbf{R}^m \rightarrow \mathbf{R}$, $A \in \mathbf{R}^{m \times n}$, and $b \in \mathbf{R}^m$. Define $h : \mathbf{R}^n \rightarrow \mathbf{R}$ by $h(x) = g(Ax + b)$

The second derivative of h is

$$\nabla^2 h(x) = A^T \nabla^2 g(Ax + b) A$$

example A.4: Find the Hessian of $h : \mathbf{R}^n \rightarrow \mathbf{R}$

$$h(x) = \log \sum_{i=1}^m \exp(a_i^T x + b_i)$$

where $a_1, \dots, a_m \in \mathbf{R}^n$, and $b_1, \dots, b_m \in \mathbf{R}$

Hint: For $g(y) = \log(\sum_{i=1}^m \exp y_i)$, we have

$$\begin{aligned} \nabla g(y) &= \frac{1}{\sum_{i=1}^m \exp y_i} \begin{bmatrix} \exp y_1 \\ \vdots \\ \exp y_m \end{bmatrix} \\ \nabla^2 g(y) &= \mathbf{diag}(\nabla g(y)) - \nabla g(y) \nabla g(y)^T \end{aligned}$$

solution. using the chain rule for composition with affine function

$$\nabla^2 h(x) = A^T \left(\frac{1}{\mathbf{1}^T z} \mathbf{diag}(z) - \frac{1}{(\mathbf{1}^T z)^2} z z^T \right) A$$

where $z_i = \exp(a_i^T x + b_i)$, $i = 1, \dots, m$

Unconstrained minimization

$$\text{minimize } f(x)$$

- f convex, twice continuously differentiable (hence $\text{dom } f$ open)
- we assume optimal value $p^* = \inf_x f(x)$ is attained (and finite)

unconstrained minimization methods

- produce sequence of points $x^{(k)} \in \text{dom } f$, $k = 0, 1, \dots$ with

$$f(x^{(k)}) \rightarrow p^*$$

- can be interpreted as iterative methods for solving optimality condition

$$\nabla f(x^*) = 0$$

Descent methods

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

- other notations: $x^+ = x + t\Delta x$, $x := x + t\Delta x$
- Δx is the *step*, or *search direction*; t is the *step size*, or *step length*
- from convexity, $f(x^+) < f(x)$ implies $\nabla f(x)^T \Delta x < 0$
(*i.e.*, Δx is a *descent direction*)

General descent method.

given a starting point $x \in \text{dom } f$.

repeat

1. Determine a descent direction Δx .
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- Gradient descent method

- $\Delta x := -\nabla f(x)$

given a starting point $x \in \text{dom } f$, tolerance $\epsilon > 0$.

repeat

1. $\Delta x := -\nabla f(x)$.

2. *Stopping criterion.* **quit** if $\|\nabla f(x)\|_2 \leq \epsilon$.

3. *Line search.* Choose step size t via exact or backtracking line search.

4. *Update.* $x := x + t\Delta x$.

- very simple, greedy algorithm

- convergence result: for strongly convex f ,

$$f(x^{(k)}) - p^* \leq c^k (f(x^{(0)}) - p^*)$$

- often very slow; rarely used in practice

- Newton Method

- $\Delta x_{\text{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x)$
- $\lambda(x) := (\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x))^{1/2}$

given a starting point $x \in \text{dom } f$, tolerance $\epsilon > 0$.

repeat

1. *Compute the Newton step and decrement.*

$$\Delta x_{\text{nt}} := -\nabla^2 f(x)^{-1} \nabla f(x); \quad \lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x).$$

2. *Stopping criterion. quit* if $\lambda^2/2 \leq \epsilon$.

3. *Line search.* Choose step size t by backtracking line search.

4. *Update.* $x := x + t\Delta x_{\text{nt}}$.

- $x + \Delta x_{\text{nt}}$ minimizes second order approximation,
 $\hat{f}(x + v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$
- $x + \Delta x_{\text{nt}}$ solves the linearized optimality condition
 $\nabla f(x + v) \approx \nabla \hat{f}(x + v) = \nabla f(x) + \nabla^2 f(x) v$
- $\lambda(x)$ is affine invariant (unlike $\|\nabla f(x)\|_2$)
- directional derivative in the Newton direction:
 $\nabla f(x)^T \Delta x_{\text{nt}} = -\lambda(x)^2$

Approximate TV de-noising

- TV denoising, a bicriterion optimization problem

$$\text{minimize } \|x - x^{\text{cor}}\|_2 + \mu \sum_{i=1}^{n-1} |x_{i+1} - x_i|$$

- $x^{\text{cor}} \in \mathbf{R}^n$ is the corrupted signal and $x \in \mathbf{R}^n$ is the de-noised signal to be computed
- Problem can be formulated as an SOCP or a QP
- *Approximate* TV denoising

$$\text{minimize } \|x - x^{\text{cor}}\|_2^2 + \mu \phi_{\text{atv}}$$

where

$$\phi_{\text{atv}}(x) = \sum_{i=1}^{n-1} \left(\sqrt{\epsilon^2 + (x_{i+1} - x_i)^2} - \epsilon \right)$$

- Objective ($\psi(x)$) is twice differentiable
- Can solve this problem using Newton's method
- Gradient and Hessian are

$$\nabla\psi(x) = 2(x - x^{\text{cor}}) + \mu\nabla\phi_{\text{atv}}(x), \quad \nabla^2\psi(x) = 2I + \mu\nabla^2\phi_{\text{atv}}(x)$$

- Use chain rule to compute $\nabla\phi_{\text{atv}}(x)$ and $\nabla^2\phi_{\text{atv}}(x)$
- Let $f : \mathbf{R} \rightarrow \mathbf{R}$ denote the function

$$f(u) = \sqrt{\epsilon^2 + u^2} - \epsilon$$

- Its first and second derivatives are

$$f'(u) = u(\epsilon^2 + u^2)^{-1/2}, \quad f''(u) = u^2(\epsilon^2 + u^2)^{-3/2}$$

- Define $F : \mathbf{R}^{(n-1)} \rightarrow \mathbf{R}$ as

$$F(u_1, \dots, u_{n-1}) = \sum_{i=1}^{n-1} f(u_i)$$

- Its gradient and Hessian are

$$\nabla F(u) = (f'(u_1), \dots, f'(u_{n-1})), \quad \nabla^2 F(u) = \mathbf{diag}(f''(u_1), \dots, f''(u_{n-1}))$$

- We have $\phi_{\text{atv}}(x) = F(Dx)$, where

$$D = \begin{bmatrix} -1 & 1 & & & & \\ & -1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & -1 & 1 & \\ & & & & & \end{bmatrix} \in \mathbf{R}^{(n-1) \times n}$$

- Using the chain rule

$$\nabla \phi_{\text{atv}}(x) = D^T \nabla F(Dx), \quad \nabla^2 \phi_{\text{atv}}(x) = D^T \nabla^2 F(Dx) D$$

- Therefore

$$\begin{aligned} \nabla \psi(x) &= 2(x - x^{\text{cor}}) + \mu D^T \nabla F(Dx) \\ \nabla^2 \psi(x) &= 2I + \mu D^T \nabla^2 F(Dx) D \end{aligned}$$

- Hessian is tridiagonal
- We can compute the Newton step in $O(n)$ operations

Matlab code for Newton method:

```
% Newton method for approximate total variation de-noising  
D = spdiags([-1*ones(n,1) ones(n,1)], 0:1, n-1, n);
```

```
% Newton's method
```

```
ALPHA = 0.01;
```

```
BETA = 0.5;
```

```
MAXITERS = 100;
```

```
NTTOL = 1e-10;
```

```
x = zeros(n,1);
```

```
newt_dec = [];
```

```
for iter = 1:MAXITERS
```

```
    d = (D*x);
```

```
    val = (x-xcor)'*(x-xcor) + ...
```

```
        MU*sum(sqrt(EPSILON^2+d.^2)-EPSILON*ones(n-1,1));
```

```
    grad = 2*(x - xcor) + ...
```

```
        MU*D'*(d./sqrt(EPSILON^2+d.^2));
```

```

hess = 2*speye(n) + ...
      MU*D'*spdiags( EPSILON^2*(EPSILON^2+d.^2).^(-3/2), ...
                    0,n-1,n-1)*D;

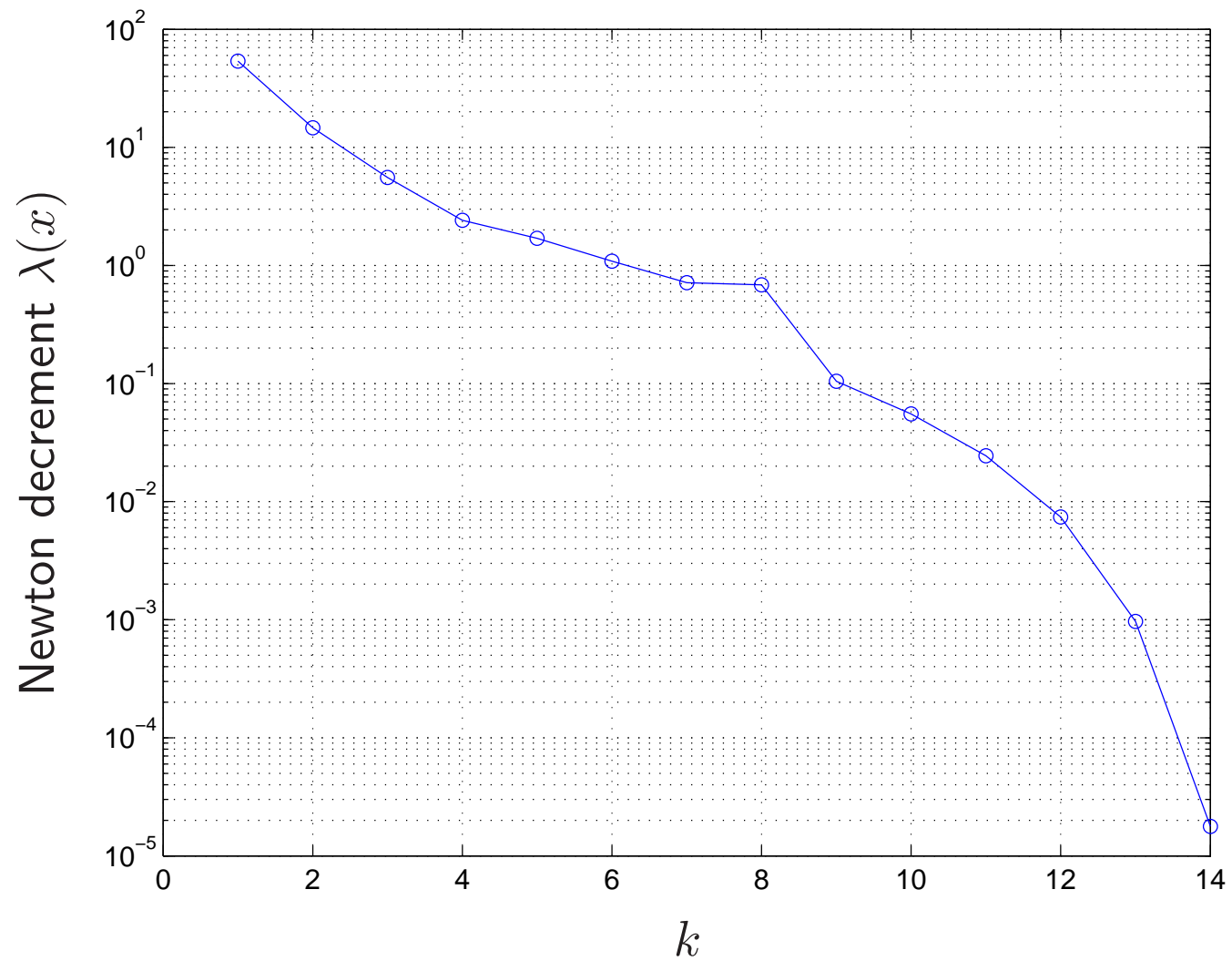
v = -hess\grad;
lambdasqr = -grad'*v;
newt_dec = [newt_dec sqrt(lambdasqr)];

if (lambdasqr/2) < NTTOL, break; end;

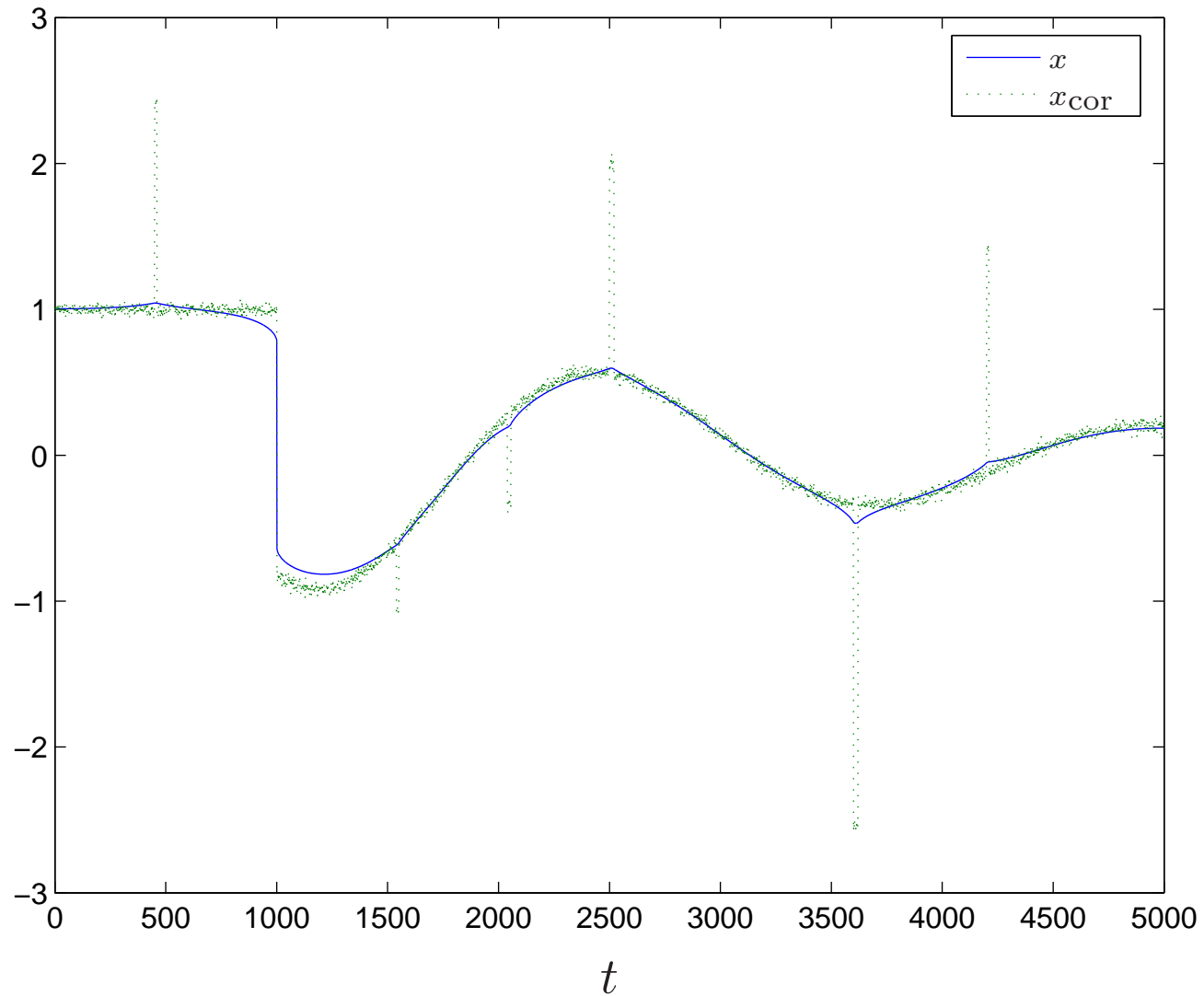
t = 1;
while ((x+t*v-xcor)'.*(x+t*v-xcor) + ...
      MU*sum(sqrt(EPSILON^2+(D*(x+t*v)).^2)-...
            EPSILON*ones(n-1,1)) > val - ALPHA*t*lambdasqr)
    t = BETA*t;
end;
x = x+t*v;
end;

```

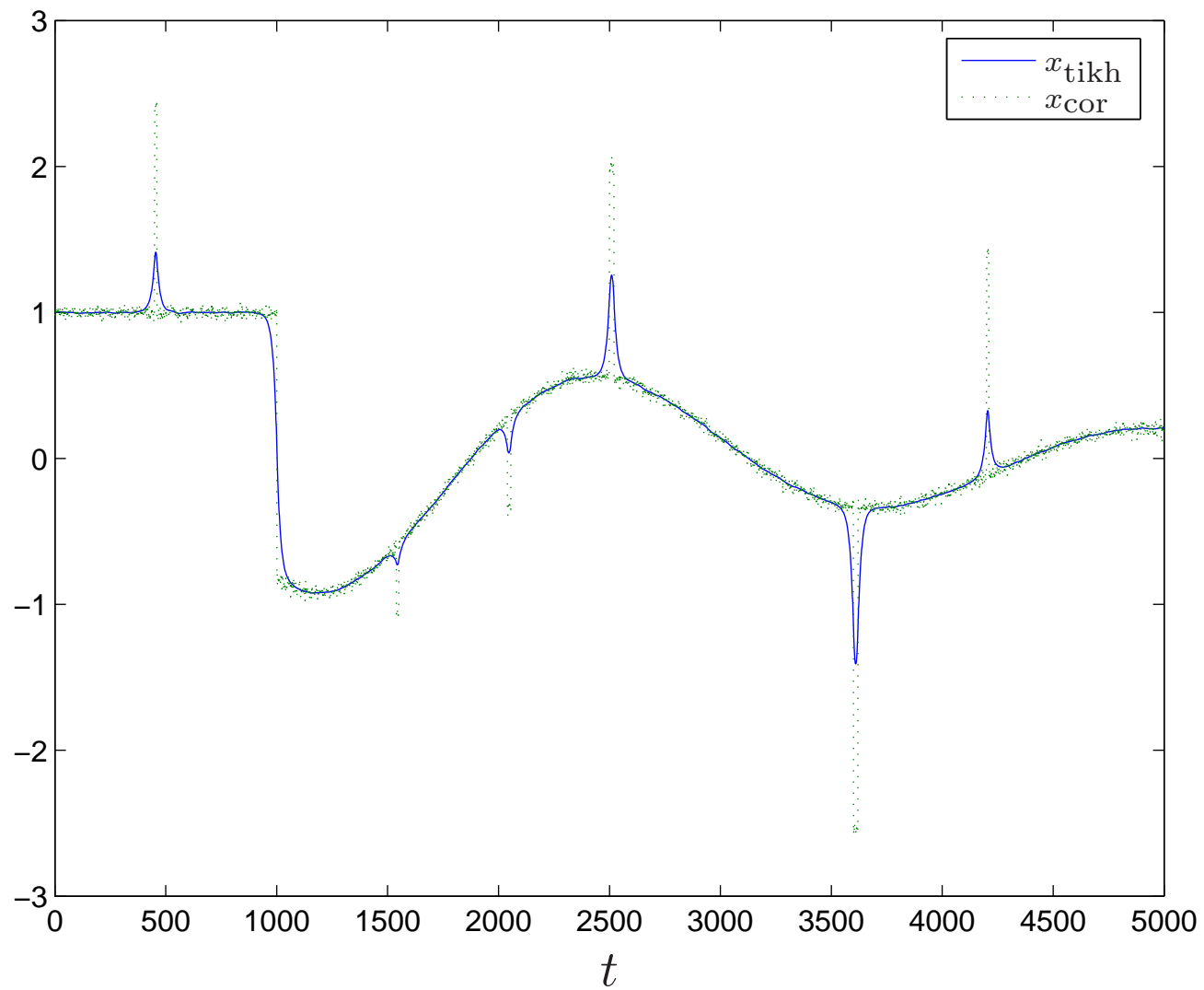
Progress of Newton's method



Approximate TV de-noising with $\epsilon = 0.001$, $\mu = 50$



Tikhonov regularized smoothing with $\mu = 250$



Tikhonov regularized smoothing with $\mu = 20000$

