Line Search Methods

Oscar Dalmau
dalmau@cimat.mx

Centro de Investigación en Matemáticas
CIMAT A.C. Mexico

February 2018

Outline

Summary

1. **Introduction**: Notation and Definitions; Norms and Matrix norms; Gradient, Hessian, Differentiation rules and Directional derivative; Taylor's formula; Big O and little o notation.
2. **Fundamentals of Unconstrained Optimization**: Introduction; Type of extrema; Necessary and Sufficient Conditions; Classification of stationary point.
3. **Convexity**: Convex sets; Convex and Concave Functions; Optimization of Convex Functions
4. **General Algorithm**: General Framework; Updating formula and Descent direction; Line search methods; Newton direction; Quasi-Newton methods; Convergence order.

General Framework

1. Start at $\boldsymbol{x}_0$, $k = 0$
2. While not converge
   - Find $\boldsymbol{x}_{k+1}$ such that $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$
   - $k = k + 1$
3. Return $\boldsymbol{x}^* = \boldsymbol{x}_k$

General Framework

1. How to choose $\boldsymbol{x}_0$?
2. Find a convergence or stop criteria?
3. How to update $\boldsymbol{x}_{k+1}$?

Updating formula

The algorithm chooses a direction $\boldsymbol{d}_k$ and searches along this direction from the current iterate $\boldsymbol{x}_k$ for a new iterate with a lower function value (line search strategy).

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{d}_k$$

Descent direction

Definition 1.1

A descent direction is a vector $\boldsymbol{d} \in \mathbb{R}^n$ such that $f(\boldsymbol{x} + t\boldsymbol{d}) < f(\boldsymbol{x})$, $t \in (0, T)$ i.e., allows to move a point $\boldsymbol{x}$ closer towards a local minimum $\boldsymbol{x}^*$ of the objective function $f : \mathbb{R}^n \to \mathbb{R}$.

There are several methods that compute descent directions, for example: use gradient descent, conjugate gradient method.

Descent direction

Descent direction

If $g(\boldsymbol{x})^T \boldsymbol{d} < 0$ then $\boldsymbol{d}$ is a descent direction.

Steepest-Descent Method

1. Compute the descent direction: $\boldsymbol{d}_k = -\boldsymbol{g}_k \stackrel{def}{=} -\boldsymbol{g}(\boldsymbol{x}_k)$
2. Compute the step length: $\alpha_k = \arg\min_{\alpha>0} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$
3. Update equation: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$

Steepest-Descent Algorithm

Algorithm 1 Steepest-Descent (Cauchy)

Require: $\boldsymbol{x}_0$

Ensure: $\boldsymbol{x}^*$

1: $k = 0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{x}_0)$
2: **while** $\|\boldsymbol{g}_k\| \neq 0$ **do**
3: $\quad \alpha_k = \arg\min_{\alpha>0} f(\boldsymbol{x}_k - \alpha \boldsymbol{g}_k)$
4: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k$
5: $\quad \boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{x}_{k+1})$
6: $\quad k = k + 1$
7: **end while**

Steepest-Descent Algorithm

If f is a quadratic function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$, with $\mathbf{A}$ symmetric and positive definite, then,

Algorithm 2 Steepest-Descent (Cauchy)

Require: $\boldsymbol{x}_0$
Ensure: $\boldsymbol{x}^*$

1: $k = 0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{x}_0)$
2: **while** $\|\boldsymbol{g}_k\| \neq 0$ **do**
3: $\quad \alpha_k = \frac{\boldsymbol{g}_k^T\boldsymbol{g}_k}{\boldsymbol{g}_k^T\mathbf{A}\boldsymbol{g}_k}$
4: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k\boldsymbol{g}_k$
5: $\quad \boldsymbol{g}_{k+1} = \nabla f(\boldsymbol{x}_{k+1})$
6: $\quad k = k + 1$
7: **end while**

Newton's Algorithm

Algorithm 3 Newton's Algorithm

Require: $\boldsymbol{x}_0$

Ensure: $\boldsymbol{x}^*$

1: $k = 0$, Solve $\nabla^2 f(\boldsymbol{x}_0)\boldsymbol{d}_0^N = -\nabla f(\boldsymbol{x}_0)$

2: **while** $\|\nabla f(\boldsymbol{x}_k)\| \neq 0$ **do**

3: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}^N$, note $\alpha_k = 1$

4: $\quad$ Solve $\nabla^2 f(\boldsymbol{x}_{k+1})\boldsymbol{d}_{k+1}^N = -\nabla f(\boldsymbol{x}_{k+1})$

5: $\quad k = k + 1$

6: **end while**

Outline

Step length

- In the computation of the *step length* α_k, we have a tradeoff: We should select α_k so that it gives sufficient reduction of f, and at the same time, we want to do it efficiently.
- One choice is to obtain the global minimizer of $\phi(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$, i.e. (exact line search method, Cauchy 1847)

$$\alpha_k = \arg\min_{\alpha > 0} \phi(\alpha)$$

- A more practical approach is to find an approximation of the previous optimization problem i.e. (inexact line search method Armijo 1966, Goldstein 1967): The idea is to efficiently compute a step length that achieves adequate reductions in f.

Step length strategy

The line search is done in two stages (iteratively):

- A *bracketing* phase finds an interval containing desirable step lengths,
- A *bisection* or *interpolation phase* computes a good step length within this interval.

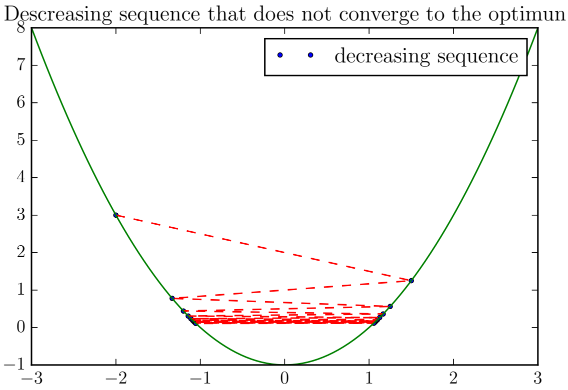
We now discuss some *termination conditions* for line search algorithms.

- A condition that we can impose on α_k, is to achieve a reduction in f, i.e., $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k)$
- The previous condition, although simple, may produce a sequence of iterates $\{\boldsymbol{x}_k\}$ where the sequence $\{f(\boldsymbol{x}_k)\}$ is decreasing but that does not converge to the optimum.

Example 2.1

1. $f(x) = x^2 - 1$. The iterates $x_k = (-1)^k(\frac{1}{k} + 1)$ yield the decreasing sequence $f(x_k) = \frac{1}{k^2} + \frac{2}{k}$ that goes to 0 when $k \to \infty$ however, the optimum is $f^* = -1$
2. $f(x) = x$. The iterates $x_k = \frac{1}{k} + 1$ yield the decreasing sequence $f(x_k) = \frac{1}{k} + 1$ that goes to 1 when $k \to \infty$. However, this function has no minimum point!

```
>>> import  numpy as np
>>> import  matplotlib.pyplot as plt
>>> x = np.linspace(-3,3,100)
>>> fx = x**2-1
>>> k = np.arange(1,20).astype('float64')
>>> xk = ((-1)**k)*(1/k+ 1) # iterates
>>> fk = 1/k**2 + 2/k  # decreasing sequence that does not
                       # converge to the optimum -1
>>> plt.plot(xk,fk, '--r', linewidth=1.0)
>>> plt.plot(xk,fk, 'ob',  markersize=3)
>>> plt.plot(x,fx, 'g')
>>> plt.savefig('noconvergence.eps', format='eps', dpi=1200)
>>> plt.show()
```


Descreasing sequence that does not converge to the optimun
decreasing sequence
8
7
6
5
4
3
2
1
0
−1
−3
−2
−1
0
1
2
3

Outline

The Wolfe Conditions

Sufficient decrease condition

An *inexact line search condition* considers that α_k should give *sufficient decrease* in the objective function f.

Armijo or sufficient decrease condition

The sufficient decrease can be measured by the following inequality

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \quad \leq \quad f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k,$$

for some constant $c_1 \in (0, 1)$.

In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$.

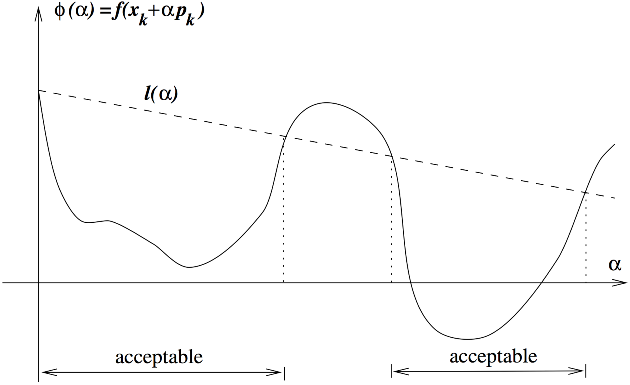
Armijo or sufficient decrease condition

$$\begin{aligned}\phi(\alpha) &= f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \\ \ell(\alpha) &= f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k\end{aligned}$$

The function $\ell(\alpha)$ has negative slope $c_1 \nabla f_k^T \boldsymbol{d}_k$. As $c_1 \in (0, 1)$, it lies above the graph of $\phi(\alpha)$ for small positive values of α.

The *sufficient decrease condition* states that α is acceptable only if $\phi(\alpha) \leq \ell(\alpha)$.

Sufficient decrease condition



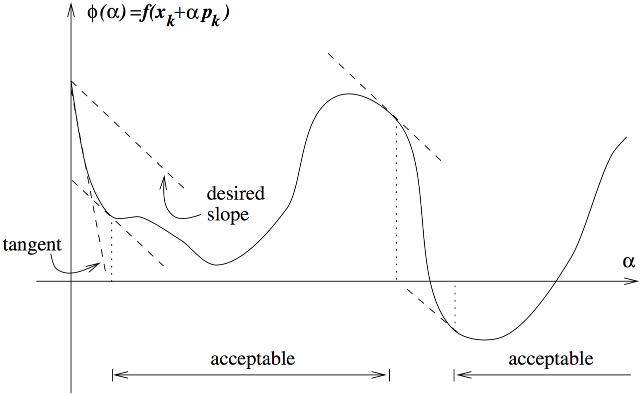
Curvature condition

- The *sufficient decrease condition* is not enough by itself to ensure that the algorithm makes reasonable progress because, it is satisfied for all sufficiently small values of α.
- In order to obtain large steps, a second requirement, called the *curvature condition*, is considered.
- The *curvature condition* says that α should satisfy

$$\nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)^T \boldsymbol{d}_k \quad \geq \quad c_2 \nabla f_k^T \boldsymbol{d}_k,$$

for some constant $c_2 \in (c_1, 1)$. A typical value is $c_2 = 0.9$.

Curvature condition



Curvature condition

Note that

$$
\begin{aligned}
\phi(\alpha) &= \nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \\
\phi'(\alpha_k) &= \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k)^T \boldsymbol{d}_k \\
\phi'(0) &= f_k^T \boldsymbol{d}_k
\end{aligned}
$$

Therefore, the curvature condition ensures that the slope of $\phi(\alpha)$ at α_k is greater than c_2 times the initial slope $\phi'(0)$, i.e.

$$
\phi'(\alpha_k) \geq c_2 \phi'(0)
$$

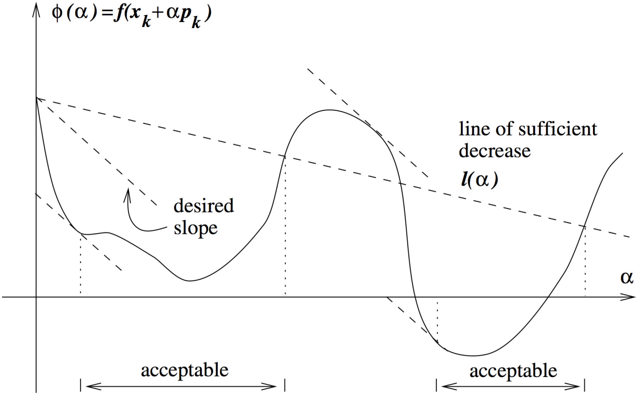
Wolfe conditions

The sufficient decrease and curvature conditions are known as the (weak) *Wolfe conditions*, i.e.

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) &\leq f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k, \\ \nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)^T \boldsymbol{d}_k &\geq c_2 \nabla f_k^T \boldsymbol{d}_k, \end{aligned}$$

with $0 < c_1 < c_2 < 1$. Typical values are $c_1 = 10^{-4}$ and $c_2 = 0.9$.

Wolfe conditions



Strong Wolfe conditions

- The step length may satisfy the *Wolfe conditions* without being close to a minimizer of $\phi()$
- We can modify the curvature condition to force α_k to lie in at least a broad neighborhood of a local minimizer or stationary point of $\phi()$.
- The *strong Wolfe conditions* require α_k to satisfy

$$\begin{aligned} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) &\leq f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k, \\ |\nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)^T \boldsymbol{d}_k| &\leq c_2 |\nabla f_k^T \boldsymbol{d}_k|, \end{aligned}$$

with $0 < c_1 < c_2 < 1$. Typical values are $c_1 = 10^{-4}$ and $c_2 = 0.9$.

Lemma 2.2

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. Let $\boldsymbol{d}_k$ be a descent direction at $\boldsymbol{x}_k$, and assume that f is bounded below along the ray $\{\boldsymbol{x}_k + \alpha \boldsymbol{d}_k | \ \alpha > 0\}$. Then if $0 < c_1 < c_2 < 1$, there exist intervals of step lengths satisfying the Wolfe conditions and the strong Wolfe conditions.

As f is bounded below and $\ell(\alpha) = f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k$ is unbounded below. Then f and ℓ intercept in a point. Let $\alpha' > 0$ the first value such that

$$f(\boldsymbol{x}_k + \alpha' \boldsymbol{d}_k) \quad = \quad f(\boldsymbol{x}_k) + c_1 \alpha' \nabla f_k^T \boldsymbol{d}_k$$

Then, the sufficient decrease condition holds for all step lengths less than α'.

Using the mean value theorem, there exists $\alpha'' \in (0, \alpha')$ such that

$$f(\boldsymbol{x}_k + \alpha' \boldsymbol{d}_k) - f(\boldsymbol{x}_k) = \alpha' \nabla f(\boldsymbol{x}_k + \alpha'' \boldsymbol{d}_k)^T \boldsymbol{d}_k$$

therefore

$$\nabla f(\boldsymbol{x}_k + \alpha'' \boldsymbol{d}_k)^T \boldsymbol{d}_k = c_1 \nabla f_k^T \boldsymbol{d}_k > c_2 \nabla f_k^T \boldsymbol{d}_k \qquad (1)$$

since $c_1 < c_2$ and $\nabla f_k^T \boldsymbol{d}_k < 0$.
Hence, by the smoothness assumption on f, there is an interval around α'' for which the Wolfe conditions hold.
Moreover, the term in the left-hand side of (1) is negative, the strong Wolfe conditions hold in the same interval.

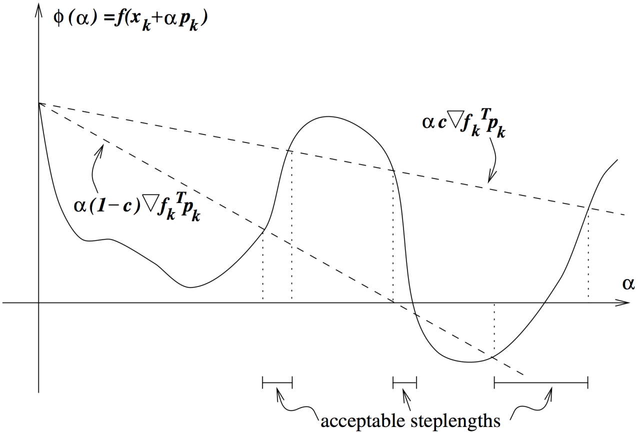
Outline

1. Like the Wolfe conditions, the Goldstein conditions ensure that the step length α achieves sufficient decrease but is not too short.
2. The Goldstein conditions can also be stated as a pair of inequalities, in the following way:

$$f(\boldsymbol{x}_k) + (1-c)\alpha\nabla f_k^T \boldsymbol{d}_k \leq f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + c\alpha\nabla f_k^T \boldsymbol{d}_k$$

with $0 < c < 1/2$.

Goldstein conditions



Outline

Sufficient decrease and Backtracking

- Choose $\hat{\alpha} > 0$, $\rho \in (0, 1)$, $c_1 \in (0, 1)$, set $\alpha = \hat{\alpha}$
- **Repeat** until $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + c_1 \alpha \nabla f_k^T \boldsymbol{d}_k$

  $\alpha = \rho\alpha$
- **end** (repeat)
- Terminate with $\alpha_k = \alpha$

Bisection with Wolfe conditions

- Choose $0 < c_1 < c_2 < 1$
- Set $\alpha = 0$, $\beta = \infty$, $\alpha^i = \alpha_0$, $i = 0$
- **Repeat**

  if $f(\boldsymbol{x}_k + \alpha^i \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + c_1 \alpha^i \nabla f_k^T \boldsymbol{d}_k$

  $\beta = \alpha^i$ and $\alpha^{i+1} = \frac{1}{2}(\alpha + \beta)$

  else if $\nabla f(\boldsymbol{x}_k + \alpha^i \boldsymbol{d}_k)^T \boldsymbol{d}_k < c_2 \nabla f_k^T \boldsymbol{d}_k$

  $$\alpha = \alpha^i \text{ and } \alpha^{i+1} = \begin{cases} 2\alpha & \text{if } \beta = \infty \\ \frac{1}{2}(\alpha + \beta) & \text{otherwise} \end{cases}$$

  otherwise

  break

  $i = i + 1$
- **end** (repeat)
- Terminate with $\alpha_k = \alpha^i$

Steepest decent with backtracking

1. Given $\boldsymbol{x}_0$, $\tau > 0$, $\rho \in (0, 1)$ and $0 < c_1 < 1$
2. Set $k = 0$
3. Compute $\boldsymbol{d}_k = -\boldsymbol{g}_k$
4. **While** (not converge), i.e. $\|\boldsymbol{g}_k\| \geq \tau$
   - Find α_k in the direction $\boldsymbol{d}_k$ with ρ, c_1 using backtracking
   - Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$
   - Compute $\boldsymbol{d}_{k+1} = -\boldsymbol{g}_{k+1}$
   - $k = k + 1$
5. Return $\boldsymbol{x}^* = \boldsymbol{x}_k$

Outline

Convergence of Line Search Methods

Let's define

$$\cos\theta_k = -\frac{\boldsymbol{g}_k^T \boldsymbol{d}_k}{\|\boldsymbol{g}_k\| \|\boldsymbol{d}_k\|}$$

where θ_k is the angle between the descent direction $\boldsymbol{d}_k$ and the steepest descent direction $-\boldsymbol{g}_k$.
The next theorem (Zoutendijk Theorem) can be used to prove the global convergence of line search algorithms (for example: steepest descent)

Zoutendijk Theorem

Zoutendijk Theorem

Consider any iteration of the form $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where $\boldsymbol{d}_k$ is a descent direction and α_k satisfies the (weak) Wolfe Conditions. Suppose that f is bounded below in $\mathbb{R}^n$ and f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} = \{\boldsymbol{x} | f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$ where $\boldsymbol{x}_0$ is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on $\mathcal{N}$, ie, there exist a constant $L > 0$ such that

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|, \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{N}$$

then

$$\sum_{k \geq 0} \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty$$

Zoutendijk Theorem: comments

- Similar results, to the previous theorem, are obtained for Goldstein conditions or Strong Wolfe Conditions
- The Zoutendijk condition implies that

$$\cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 \quad \rightarrow \quad 0 \tag{2}$$

- The previous result can be used to prove global convergence for line search algorithms

Zoutendijk Theorem: comments

- If the optimization method ensures the angle θ_k is away from 90^o, ie, there is a positive δ such that $\cos\theta_k \geq \delta > 0$ for all k. It follows from (2) that

$$\|\nabla f(\boldsymbol{x}_k)\|^2 \quad \rightarrow \quad 0 \tag{3}$$

Zoutendijk Theorem: comments

- This means that the gradient norms $\|\nabla f(\boldsymbol{x}_k)\|$ converges to zero, due to the descent directions are not too close to be orthogonal to the gradient.
- In particular, the steepest descent with the line search strategy satisfying the (weak) Wolfe Condition produces a gradient sequence that converges to zero, due to the descent direction $\boldsymbol{d}_k = -\boldsymbol{g}_k$ is parallel to the gradient.
- This guarantees the convergence to a stationary point but not to a local minimizer.

1. Implement the steepest descent algorithm using the backtracking method.
2. Obtain the minimum of the following functions with previous algorithm. Plot (k, f_k) and $(k, \|\boldsymbol{g}_k\|)$

$$
\begin{aligned}
f(x, y) &= (1-x)^2 + 100(y - x^2)^2 \\
f(\boldsymbol{x}) &= \sum_{i=1}^{n-1} (1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2
\end{aligned}
$$

3. Obtain the minimum of $f(\boldsymbol{x})$ for $\eta \sim \mathcal{N}(0, \sigma)$ and $\lambda > 0, \sigma > 0$. Plot (t_i, y_i) and (t_i, x_i^*) in the same figure.

$$
\begin{aligned}
f(\boldsymbol{x}) &= \sum_{i=1}^{n-1} (x_i - y_i)^2 + \lambda(x_{i+1} - x_i)^2 \\
y_i &= t_i^2 + \eta, \; t_i = \frac{2}{n-1}(i-1) - 1, \; i = 1, 2, \cdots, n.
\end{aligned}
$$