Optimization

Classical problems:

- find $x$ s.t. $f(x)$ is minimized

Interesting cases:

- $f \in C^2$ (cont. and 1, 2 deriv.)
- $f \in C^1$, cont
- $f \in C^0$, $f$ convex
- $f \in C^0$

Constrained: $x \in \{ u \mid g(u) = 0 \}$

Discrete: $x \in \{0, 1\}^n$
General methods:

Search:

- Construct a seq $x_i$
- $x_n \to$ right answer

Qns:
- How?
- When to stop?
- Converge how fast?

Closed form:

- Uncommon, but sometimes useful
- Write conditions that identify a solution, then find it

$\Rightarrow$ Very valuable in itself!
Cases and terminology:

Continuous optimization

- Usually, $f$ is at least continuous
- Usually, $x \in \mathbb{R}^n$

for the moment

If for $x \in B(x^*)$

(open ball centered at $x^*$)

we have

$f(x^*) < f(x)$

then $x^*$ is a local minimum

If we also have for $x \in \mathbb{R}^n$

$f(x^*) \leq f(x)$

$x^*$ is a global minimum
Some examples suggest

- There can be many $x^*$ that are a *global minimum*.
  (But they all have the same $f(x^*)$!)

- Minima *don't* have to be isolated!

![Diagram](attachment:image.png)
Some closed form examples

General idea: If at any step in any direction takes you uphill, you must be at a local min.

Most helpful when $f$ is differentiable (but more to come)

\[ f \begin{cases} \text{diff} & \text{if } \nabla f(x^*) = 0 \\ \text{not diff} & \text{otherwise} \end{cases} \]

this means

\[ f(x^* + \delta x) > f(x^*) \]

for any $\delta x$ s.t. $\|\delta x\|$ is sufficiently small.

\[ \downarrow \text{this matters!} \]
Now assume $f$ is continuously differentiable (i.e. $f$ is continuous; derivative of $f$ exists and is continuous)

$$f(x^* + \delta x) = f(x^*) + \nabla f \cdot \delta x + O(\delta x^2).$$

But at $x^*$, any direction is uphill

so \[ \nabla f = 0 \]

for small enough \[ \| \delta x \| \]

and this is useful.

Recipe 1:

$$f(x) = x^T A x + b^T x + c$$

$$\nabla f = A x + b$$

where and when is there a local minimum?

$$x \mid A x + b = 0$$

\[ \rightarrow \text{These are our suspects.} \]
But these might not be local minima.

Change coordinates:
\[ u = x + w \]
where \( Aw = b \)

Does this exist?
What if it doesn't?

Then
\[ Au = 0 \]
is our condition.

**Cases**

A has full rank

\[ \rightarrow u = 0 \]

Q: \[ u^T A s u \geq 0 \] for all small enough \( s u \)?

1) \( A \succeq 0 \), yes.
   - Positive definite;
   - means \( w^T A w > 0 \) for all \( w \neq 0 \).

2) Not isolated local min!
A does not have full rank
\[ \Rightarrow \text{not isolated local min.} \]

Remember this
\[
f(x) = x^T A x + b^T x + c
\]
has \text{isolated} local min at
\[ x \text{ st. } A x + b = 0 \]

If \( A \succeq 0 \)
has \text{isolated} local max at
\[ x \text{ st. } A x + b = 0 \]

If \( -A \succeq 0 \)
Variational:

choose \( f \) such that

\[
\int F(c^*, \Phi(c^*)) \, dc^* \text{ is minimized}
\]

How do we know we are at a local min?

- Any step in any direction makes cost get bigger
- Easy test if \( f \) is differentiable \( C^1 \)
  
  \[
  \nabla f = 0
  \]

- Life is harder if \( f \) is \( C^0 \) or worse
  
  - Local test might be difficult
  
  - \text{eg.}
we want to fill in \( \Omega \)

reasonable criterion:

\[
\min \int_{\Omega} \| \nabla f \|^2 \, dA
\]

subject to: \( f = 0 \) \( I \) on \( \partial \Omega \)

i.e. don't create derivatives unnecessarily
- agree w/ boundary.

How could we solve this?

- discrete, work with discretized derivative and integral
- we are now minimizing a function of a (big!) vector
Alternative:

- What properties does $f$ have?

$\rightarrow$ assume that $\hat{f}$ is the soln.

$\rightarrow$ now, for ANY test function $\phi$, such that $\phi = 0$ on $\partial \Omega$,
we have

$$\int_{\Omega} \| \nabla (f + \phi) \|^2 \, dA \geq \int_{\Omega} \| \nabla f \|^2 \, dA$$

for small enough $\varepsilon > 0$

(i.e. if you make a small move in any direction, the value goes up)
Now, this means
\[ d \int_{\Omega} \| \nabla f + \varepsilon \phi \|^2 \, dA = 0 \quad \text{for any } \phi \]
now this is
\[ 2 \int_{\Omega} \nabla f \cdot \nabla \phi \, dA = 0 \quad (\text{doesn't seem helpful}) \]
But recall
\[ \nabla \cdot [ \phi \nabla \cdot \phi ] = (\nabla \phi) \cdot \nabla \cdot \phi + \phi (\nabla \cdot \nabla) \]
i.e.
\[ \int_{\Omega} \nabla \cdot [\phi \nabla \cdot \phi] \, dA = \int_{\Omega} \nabla \phi \cdot \nabla f \, dA + \int_{\Omega} \phi (\nabla^2 f) \, dA \]
now
\[ \int_{\Omega} \nabla \cdot [\phi \nabla f] \, dA \]
\[ = \int_{\partial \Omega} (\phi \nabla f) \cdot ds \]
(divergence theorem - remember!)

but \( \phi = 0 \) on \( \partial \Omega \).

so \[ \int_{\Omega} \nabla \phi \cdot \nabla f \, dA = -\int_{\Omega} \phi (\nabla^2 f) \, dA = 0 \]

But this is true for any \( \phi \).

so \( \nabla^2 f = 0 \)

and this offers other ways to solve.
Give a nice criterion for variational case

\[
\frac{d}{dz} \left[ \int F(u, g(u) + \varepsilon \phi(u)) \, du \right] \bigg|_{\varepsilon=0} = 0
\]

**Variational example**

\[
\min_{g} \int_{0}^{a} \sqrt{1 + \left( \frac{dg}{du} \right)^2} \, du
\]

Subject to:

\[
g(0) = 0
\]
\[
g(a) = b
\]
i.e. for any test function $\varphi$, we know $g$ is right if

$$\frac{d}{ds} \left[ \int \frac{1}{1 + \left( g + \epsilon \varphi \right)^2} \, du \right] \bigg|_{s=0} = 0$$

now write $\frac{dg}{du} = g'$, $\frac{d\epsilon \varphi}{du} = \epsilon' \varphi'$

i.e.

criterion is:

$$\int \frac{g' \varphi'}{\sqrt{1 + g'^2}} \, du = 0$$

Not promising; but use integration by parts

(i.e. $\int u v' \, dx = uv \bigg|_{x=a}^{x=b} - \int v u' \, dx$)
to get
\[ \int \frac{d}{du} \left[ \frac{g'}{(1 + g'^2)^{1/2}} \right] \cdot \varphi \, du = 0 \]
for any \( \varphi \) \( (\varphi(0) = \varphi(a) = 0) \).

This gives \( \frac{d}{du} \left[ \frac{g'}{(1 + g'^2)^{1/2}} \right] = 0 \),
which means
\[ g'' \left[ \frac{1 - \frac{g'^2}{(1 + g'^2)^{1/2}}}{(1 + g'^2)^{1/2}} \right] = 0 \]
and \( \ldots \) is always positive so \( g'' = 0 \).

(The shortest distance between 2 points is a line.)

Equations obtained like this are Euler-Lagrange equations.
Variational problems can be quite delicate — a solution could not exist in reasonable function spaces.

Example

\[
\min_f \int_0^1 \left( \theta f'(u) + \left[ \left( \frac{df}{du} \right)^2 - 1 \right]^2 \right) du = 7 \left[ f \right]
\]

s.t. \( f(0) = 0 \); \( f(1) = 0 \).

Solution looks like \( \theta \)

\[
\begin{array}{c}
\text{df} = -1 \\
\text{du} \\
\text{df} = 1 \\
\text{du}
\end{array}
\]

but as \( \theta \) gets smaller, we have

what \( f \) gets smaller, too

\( \rightarrow \) but there can’t be a limit

So no solution.
Failure of a solution to exist occurs in practical problems.

Sample issues:

$$\arg \min_u \ u^2 \quad u \in (0, 1]$$

(open)

(This doesn't happen all that often, but is worth keeping in mind.)

Harder issues:

$$\arg \min_{f} \ \int_0^1 \left[ \frac{1}{\sqrt{1+(f')^2}} - \frac{1}{\sqrt{2}} \right]^2 \ dx$$

S.t. \ \ f(0) = 0

\ \ f(1) = 0
This sort of thing turns up in shape from shading problems rather often.

Notice I can get a min of the objective if $f'{}^2 = 1$.

$\rightarrow$ So, if $f \in C^0$ no solution (there can't be a function $f$ such that $f'{}^2 = 1$, $f(0) = 0$, $f(1) = 0$).

$\rightarrow$ If $f \in C^0$, too many solutions! \\
\[ ^\wedge,的精神，\wedge, \wedge, \text{etc.} \]

Now assume that
In practice, we usually turn variational problems into continuous optimization problems by writing

\[ f = \sum_{i} a_{i} g_{i} \]

basis functions

then solving for \( a_{i} \)

**But**

- Bad stuff can happen if original problem is poorly set
  - The reasoning comes in useful later

Crucial, Take Home point:

You are at a minimum if every available step is uphill
Now consider:

$$\min_{f(x)} \quad f \in C^2$$

A descent direction $d$ has the property that

$$f(x_0 + \varepsilon d) < f(x_0) \quad \text{for} \quad \varepsilon < r_d$$
Both gradient descent and coordinate descent are **naughty**.

- Local model of function as quadratic form

\[ \nabla f \text{ is a level curve}\]

**Best step** is not along gradient.

**Best step**
There are numerous descent directions

\[ d_g = -\frac{\nabla f}{\|\nabla f\|} \]

This is gradient descent

Issues:
- how to choose \( \epsilon \)
- perhaps interval halving
- more sophisticated machinery later

Write \( P_c \) for projection to some set of coordinate axes
- i.e. \( P_c \) zeros some elements

\[ d_{cd} = -P_c d_g \]

This is coordinate descent
Both gradient descent and coordinate descent are 
naughty.

- Local model of function as quadratic form

\[ f(x) = a \]

level curve

gradient, perp to level curve

best step is not along gradient

\[ \nabla f(x) \]

best step
if we take the best step along the gradient in this case, we don't go far along axis of symmetry.

Now we fitting slowly down the axis.

Notice, best step can be at (-90°, 90°) to gradient.

function is:

\[ x' A x \], \quad A \text{ positive definite}

we are at \( u \).

gradient is: \[ 2 A u \]

best step \( \Delta x \) is: \[ -u \].
so the angle is:

$$\frac{\mathbf{u}'\mathbf{A}\mathbf{u}}{(\mathbf{u}'\mathbf{u})^{1/2} (\mathbf{u}'\mathbf{A}'\mathbf{A}\mathbf{u})^{1/2}}$$

Coordinate descent is also naughty.
Newton's method:

\[ f(x_0 + s) \approx f(x_0) + \nabla f \cdot s + \frac{1}{2} s^T H_f s + O(s^3) \]

we could minimize the quadratic part as a function of \( s \):

\[ \min_s \nabla f \cdot s + \frac{1}{2} s^T H_f s \]

i.e.

\[ \nabla f + H_f s = 0 \]

\[ H_f s = -\nabla f \]
Q: Does Newton’s method always give a descent direction?

\[
f(x + d) = f(x) + \nabla f d + \frac{1}{2} d' H f d + O(d^3)
\]

\[
d = -H_f^{-1} \nabla f
\]

\[
f(x + d) - f(x) \approx \frac{1}{2} \nabla f H_f^{-1} \nabla f
\]

- So we’re ok if $H_f$ is positive definite.

Q: is $p_k$ a descent direction?

A: if $p_k' \nabla f_k < 0$

Notice we can obtain descent dirs from

\[
p_k = -B_k^{-1} \nabla f_k
\]

$p_k$ iteration.
\[ B_k^{-1} = 1d \quad \text{gradient} \]
\[ B_k^{-1} = p \quad \text{coord} \]
\[ B_k^{-1} = H_f \quad \text{Newton} \]

But

\[ B_k \text{ must be P.d.f for } \theta \text{ to be a descent} \]

Example: coordinate descent and EM

we have two parametric models

\[ p(x/\theta_1) = e^{-g(x;\theta_1)} \]

and horse model

\[ p(x/\theta_2) = e^{-g(x;\theta_2)} \]

and we observe \( x_i \) from a mixture

\[ p(x/\theta) = \mu_1 p_1 + \mu_2 p_2 \]
**Expectation - Maximization**

- Assume we have a mixture model
  
  \[ P(x | \theta) = \mu, P(x | \theta_1) + (1-\mu) P(x | \theta_2) \]

- For simplicity, I'll work with a mixture of exponentials
  
  \[ P_1(x | \theta_1) = e^{-g_1(x, \theta_1)} \]
  
  \[ P_2(x | \theta_2) = e^{-g_2(x, \theta_2)} \]

- I now have
  
  \[ x_1, \ldots, x_n \sim P(x | \theta) \]

  \( \rightarrow \) What is \( \theta = (\theta_1, \theta_2, \mu) \)?

- Inference by maximum likelihood will be hard because we must find
  
  \[ \arg \max_{\theta} \sum_{i} \log \left[ \mu e^{-g_1(x_i, \theta_1)} + (1-\mu) e^{-g_2(x_i, \theta_2)} \right] \]
This is difficult to work with, so

The problem can be simplified if we know the mixture component from which each $x_i$ comes. Let $S_i = \{0, 1, 2\}$

This gives

i.e. $P(x_i | S_i, \Theta)$ is straightforward

Write the complete data log-likelihood

$$
\ell_c(\Theta) = \sum_i \log \left[ P(x_i, S_i | \Theta) \right] 
$$

$$
= \sum_i \left[ \log P(x_i | S_i, \Theta) + \log P(S_i | \Theta) \right]
$$
General algorithm:

\[ h(\theta) = F(s, \theta) \]

we want to estimate these

we don't know these.

assume we have an \text{estimate} \( \theta^{(n)} \) of \( \theta \)

then

\[ Q(\theta; \theta^{(n)}) = E_{s|\theta^{(n)}} \left[ F(s, \theta) \right] \]

function of \( \theta \) that depends on \( \theta^{(n)} \)

\[ \text{E-Step:} \quad \text{Form} \quad Q(\theta; \theta^{(n)}) \]

\[ \text{M-Step} \quad \theta^{(n+1)} = \arg \max_{\theta} Q(\theta; \theta^{(n)}) \]
In our case

\[ L_c(\Theta) = \sum_{i} \left[ -\delta_i g_1(x_i, \Theta) + (1-\delta_i) g_2(x_i, \Theta) \right] \]

\[ + \sum_{i} \left[ \delta_i \log \mu + (1-\delta_i) \log (1-\mu) \right] \]

\[ E_{S|x,\hat{\Theta}} L_c(\Theta) \text{ is straightforward, because } L_c(\Theta) \text{ is linear in } S. \]

\[ \rightarrow \text{ Substitute } P(S_i=1|x, \hat{\Theta}) \text{ for } S_i. \]

\[ P(S_i=1|x, \hat{\Theta}) = \frac{P(x_i, S_i=1|\hat{\Theta})}{P(x_i|\hat{\Theta})} \]

\[ = \frac{P(x_i|S_i=1, \hat{\Theta}) P(S_i=1|\hat{\Theta})}{P(x_i|S_i=1, \hat{\Theta}) P(S_i=1|\hat{\Theta}) + P(x_i|S_i=0, \hat{\Theta}) P(S_i=0|\hat{\Theta})} \]
In our case

\[
P(\delta_i = 1 | x_i, \hat{\theta}^{(i)}) = \frac{e^{-g_1(x_i, \hat{\theta}^{(i)}) \mu}}{e^{-g_1(x_i, \hat{\theta}^{(i)}) \mu} + e^{-g_2(x_i, \delta^{(i)}) \mu + \epsilon}}
\]

Now, consider the following objective fn.

\[
I(\theta, s) = I_c(\theta) + H(s)
\]

Entropy

\[
- \sum_i \left[ \delta_i \log \delta_i + (1 - \delta_i) \log (1 - \delta_i) \right]
\]
we will do coordinate descent

- fix $\theta$, min wrt $s$
- fix $s$, min wrt $\theta$

Consider

$$\nabla_s F^{\text{1st}}(\theta, s) = 0$$

$$\frac{\partial F}{\partial s_i} = -\left[ \log s_i - \log(1-s_i) \right]$$

$$+ \left[ -g_i + \log \hat{\mu} \right]$$

$$+ \left[ g_2 - \log(1-\mu) \right]$$

$$= 0$$
i.e. 
\[ \frac{S_i}{1 - S_i} = \frac{e^{-g_1 \mu}}{e^{-g_2 (1 - \mu)}} \]

and we substitute these in.

But this is what E step does.

\[ \mu \text{ step } = \nabla_\mu f(\theta, S_i) = 0 \]

Q: Why not do Newton?

A: Not sure frankly.

H is big but sparse?
\[ \frac{s_i}{1 - s_i} = \frac{e^{-\mu}}{e^{-\beta} (1 - \mu)} \]

hence: ELM is coordinate ascent

Q: why not do reparameter method?
A: not sure, frankly.

H is big, but sparse

Issues:

- What to do with a descent dir?
- How to make H behave?
  \[ \rightarrow \text{big} \]
  \[ \rightarrow \text{not P.D.} \]
- How bad is gradient descent?
We have $p_k$, and wish to choose a step length $\alpha$.

Consider $f(x_k + \alpha p_k) > 0$.

What $\alpha$ are acceptable?

- Ideally, $\alpha$ to global minimizer
- Sufficient decrease

\[ f(x_k + \alpha p_k) < f(x_k) + c_1 \alpha \nabla f(x_k)^T p_k \]

\[ 0 < c_1 < 1 \]

for some constant (typically $1e^{-4}$)

**Armiijo Condition**

**Wolfe**

\[ f(x_k + \alpha p_k) \]
Sufficient decrease is not enough
- very small $\alpha$ are OK.

\[
\forall \epsilon > 0 \quad \nabla f(x_k + \alpha p_k) p_k \geq c_2 \nabla f(x_k) p_k
\]

$c_1 < c_2 < 1$

Notice:

Write $\Phi(\alpha) = f(x_k + \alpha p_k)$

Then

\[
\frac{d\Phi}{d\alpha} = \nabla f(x_k + \alpha p_k) p_k
\]

So condition is:

\[
\frac{d\Phi}{d\alpha} \leq c_2 \nabla f(x_k) p_k = c_2 \frac{d\Phi}{d\alpha} \bigg|_{\alpha=0}
\]

Notice sign of slope!
$c_2$ is usually $0.4$ (Newton) $0.1$ (conj. grad.)

Wolfe conditions

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k$$

$$\nabla f(x_k + \alpha_k p_k) k \geq c_2 \nabla f_k^T p_k$$

Notice: for $f$ continuously diff., $f$ bounded below along $x_k + \alpha_k p_k$, $\alpha > 0$ there exist intervals satisfying these coward.

Alg:

for $\alpha > 0$, $\rho \in (0, 1)$

$d = \alpha$

repeat until [sufficient descent]

end

OK for Newton; not as good for others.
Now we are generating a sequence \( \{x_i\} \) by finding \( p_k, \alpha_k \) and accepting

\[
x_{k+1} = x_k + \alpha_k p_k
\]

Q: How does this set behave?
A:

Q: To what does it converge?
Q: How fast?
Some answers by defining

\[ \cos \theta_k = -\frac{\nabla f_k^T P_k}{\| \nabla f_k \| \| P_k \|} \]

Thus: (Zoutendijk)

Consider iteration of given form, \( x_k \)
satisfying Wolfe condi ons, \( f \) bounded below
continuously df. In an open set \( \mathcal{N} \)
containing \( x_0 \), let \( \mathcal{N} = \{ x : f(x) \leq f(x_0) \} \).
Assume \( \nabla f \) is Lipschitz. Then

\[ \sum_k \cos^2 \theta_k \| \nabla f_k \|^2 < \infty \]
Rates of convergence:

- We have, for $f \in C^2$, exact line search (i.e., best $x_k$), $x^*$ the min,
  Gradient descent behaves like

\[ f(x_{k+1}) - f(x^*) \leq \frac{r^2}{2} \left[ f(x_k) - f(x^*) \right], \]

related to Hessian

- For Newton, if $x_0$ sufficiently close to $x^*$

\[ \| x_{k+1} - x^* \| \leq \frac{1}{2} \| x_k - x^* \|^2 \]

related to Hessian
Newtons method with Classical modification

Problem: \( H \) may not be PD, so \( H_k p = -\nabla f \) may not give a descent direction.

Strategy: modify \( H \) to be PD.

\[ B_k = H_k + E_k \]

\( E_k \) chosen to make \( B_k \) PD

This will converge globally if

\[ E_k \neq \{ H_k (x_k) \} \] is bounded

\[ \Rightarrow \| B_k \| \| B_k^{-1} \| \leq C > 0 \]
Generally would like $K \rightarrow$ small
(To as to preserve Hessian info)

1: Add a multiple of Identity:

\[ \text{if } \min \{ h_{ii} \} > 0 \]
\[ C_0 = 0 \]
\[ \text{else} \]
\[ C_0 = -\min (h_{ii}) + \beta ; \]
\[ \text{end} \]

for $k =$ \[ \ldots \]

attempt Cholesky factorization of $H + \gamma_k I$
if OK return factor
\[ \text{else} \]
\[ \gamma_{k+1} = \max (2\gamma_k, \beta) \]
\[ \text{end} \]
\[ \text{end} \]

\[ \text{we are searching for } \gamma I \text{ to make } H \text{ p.d.} \]
Cholesky:

\[ A = LL^T \]

- works if \( A \) PD, otherwise

get a \( \sqrt{0}, \sqrt{-ve} \)

Modified Cholesky

\[ A = LDL^T \]

Where

\( L \) is lower triangle, \( D \) is diag, \( +ve \) diag

Note: if \( A \) pd, then \( D \) elements are +ve
Cholesky:

for \( j = 1 \ldots n \)

\[
c_{jj} = a_{jj} - \sum_{s=1}^{j-1} d_{s} e_{js}^2
\]
\( d_j = c_{jj} \)

for \( i = j+1 \ldots n \)

\[
c_{ij} = a_{ij} - \sum_{s=1}^{j-1} d_{s} e_{is} e_{js}
\]
\( e_{ij} = c_{ij} / d_j \)

end
end

Now: \( d_{jj} \) all positive if \( A \) PD.

Modify alg so that

\[
d_{ij} = \max \left( 1, \frac{|c_{ij}|}{\sqrt{\sum_{j} \Theta_{jj}^2}} \right)
\]

\[
\Theta_{jj} = \max_{j < i < n} |c_{ij}|
\]
and this gives a "factorization"

where

\[ d_i \gg s \]

\[ |m_{ij} = e_{ij}\sqrt{d_j}| < s \]

Desirable for error control.

-. Improvements

-. Permute rows and columns to reduce the size of the modification.

-. This will give guaranteed bounds of global convergence.
Step length selection:

$$\Phi(x) = \Phi(x_0 + \alpha p_k)$$

Sufficient decrease is then:

$$\Phi(x) \leq \Phi(0) + c, \alpha \Phi'(0).$$

Guess \(x_0\)

\(\rightarrow\) OK; stop

\(\rightarrow\) Not OK, there is an OK step in interval.

- We know \(\Phi(0), \Phi(x_0), \Phi'(0)\)
- Build quadratic interpolate

$$\Phi(x) = \left( \frac{\Phi(x_0) - \Phi(0) - x_0 \Phi'(0)}{x_0^2} \right) x^2 + \Phi'(0) x + \Phi(0)$$

Minimize in \(\alpha\) to get \(x_1\)
\[ \rightarrow \alpha, \text{OK} \; ; \; \text{stop} \]

\[ \rightarrow \text{else construct cubic} \]

\[ \text{interpolate of } \varphi(0), \varphi'(0), \varphi'(\alpha_0), \varphi(\alpha_1) \]

\[ \text{minimize } \alpha_2 \]

\[ \rightarrow \alpha_2, \text{OK} \; \text{stop} \]

\[ \rightarrow \text{else cubic with } \varphi(0), \varphi'(0), \text{two most recent } \alpha \]
It can be shown that if \( x_k \to x^* \) superlinearly, then the ratio in this expression converges to 1. If we adjust the choice (3.60) by setting

\[
\alpha_0 \leftarrow \min(1, 1.01 \alpha_0),
\]

we find that the unit step length \( \alpha_0 = 1 \) will eventually always be tried and accepted, and the superlinear convergence properties of Newton and quasi-Newton methods will be observed.

**A LINE SEARCH ALGORITHM FOR THE WOLFE CONDITIONS**

The Wolfe (or strong Wolfe) conditions are among the most widely applicable and useful termination conditions. We now describe in some detail a one-dimensional search procedure that is guaranteed to find a step length satisfying the strong Wolfe conditions (3.7) for any parameters \( c_1 \) and \( c_2 \) satisfying \( 0 < c_1 < c_2 < 1 \). As before, we assume that \( p \) is a descent direction and that \( f \) is bounded below along the direction \( p \).

The algorithm has two stages. This first stage begins with a trial estimate \( \alpha_1 \), and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths. In the latter case, the second stage is invoked by calling a function called zoom (Algorithm 3.6, below), which successively decreases the size of the interval until an acceptable step length is identified.

A formal specification of the line search algorithm follows. We refer to (3.7a) as the **sufficient decrease condition** and to (3.7b) as the **curvature condition**. The parameter \( \alpha_{\max} \) is a user-supplied bound on the maximum step length allowed. The line search algorithm terminates with \( \alpha_* \) set to a step length that satisfies the strong Wolfe conditions.

**Algorithm 3.5 (Line Search Algorithm).**

Set \( \alpha_0 \leftarrow 0 \), choose \( \alpha_{\max} > 0 \) and \( \alpha_1 \in (0, \alpha_{\max}) \);

\[
i \leftarrow 1;
\]

**repeat**

Evaluate \( \phi(\alpha_i) \);

\[
\text{if } \phi(\alpha_i) > \phi(0) + c_1 \alpha_i \phi'(0) \text{ or } [\phi(\alpha_i) \geq \phi(\alpha_{i-1}) \text{ and } i > 1] \]

\[
\alpha_* \leftarrow \text{zoom}(\alpha_{i-1}, \alpha_i) \text{ and stop};
\]

Evaluate \( \phi'(\alpha_i) \);

\[
\text{if } |\phi'(\alpha_i)| \leq -c_2 \phi'(0) \]

\[
\text{set } \alpha_* \leftarrow \alpha_i \text{ and stop};
\]

\[
\text{if } \phi'(\alpha_i) \geq 0 \]

\[
\text{set } \alpha_* \leftarrow \text{zoom}(\alpha_i, \alpha_{i-1}) \text{ and stop};
\]

Choose \( \alpha_{i+1} \in (\alpha_i, \alpha_{\max}) \);

\[
i \leftarrow i + 1;
\]

**end (repeat)**

**Algorithm 3.6 (zoom).**

**repeat**

Interpolate (using a trial step \( \alpha_{\trial} \));

Evaluate \( \phi(\alpha_{\trial}) \);

\[
\text{if } \phi(\alpha_{\trial}) > \phi(0) \]

\[
\alpha_{\hi} \leftarrow c \alpha_{\max};
\]

**else**

Evaluate \( \phi'(\alpha_{\trial}) \);

\[
\text{if } |\phi'(\alpha_{\trial})| \leq -c_2 \phi'(0) \]

\[
\text{set } \alpha_* \leftarrow \alpha_{\trial} \text{ and stop};
\]

\[
\text{if } \phi'(\alpha_{\trial}) \geq 0 \]

\[
\text{set } \alpha_* \leftarrow \text{zoom}(\alpha_{\trial}, \alpha_{\trial-1}) \text{ and stop};
\]

Choose \( \alpha_{i+1} \in (\alpha_{\trial}, \alpha_{\max}) \);

\[
i \leftarrow i + 1;
\]

**end (repeat)**

Note that the sorting of the arguments of the function zoom is important that the successful function call requires a finite number of iterations.

We now specify its input arguments:

1. the interval bounds \( [\alpha_{\lo}, \alpha_{\hi}] \);
2. the strong Wolfe conditions;
3. \( \alpha_{\lo} \) is, among all acceptable step lengths, the one we wish to use.

Each iteration of zoom proceeds by bisecting the interval defined by these endpoints by \( \alpha_{\mid} \) and setting

\[
\alpha_{\mid} = \text{zoom}(\alpha_{\lo}, \alpha_{\hi});
\]

where $\text{zoom}(\alpha_{\lo}, \alpha_{\hi})$ is the midpoint of the interval $[\alpha_{\lo}, \alpha_{\hi}]$. The algorithm continues until an acceptable step length is found.
Note that the sequence of trial step lengths \( \{\alpha_i\} \) is monotonically increasing, but that the order of the arguments supplied to the zoom function may vary. The procedure uses the knowledge that the interval \((\alpha_{i-1}, \alpha_i)\) contains step lengths satisfying the strong Wolfe conditions if one of the following three conditions is satisfied:

(i) \( \alpha_i \) violates the sufficient decrease condition;
(ii) \( \phi(\alpha_{i}) \geq \phi(\alpha_{i-1}) \);
(iii) \( \phi'(\alpha_{i}) \geq 0 \).

The last step of the algorithm performs extrapolation to find the next trial value \( \alpha_{i+1} \). To implement this step we can use approaches like the interpolation procedures above, or we can simply set \( \alpha_{i+1} \) to some constant multiple of \( \alpha_i \). Whichever strategy we use, it is important that the successive steps increase quickly enough to reach the upper limit \( \alpha_{\text{max}} \) in a finite number of iterations.

We now specify the function zoom, which requires a little explanation. The order of its input arguments is such that each call has the form \( \text{zoom}(\alpha_{\text{lo}}, \alpha_{\text{hi}}) \), where

(a) the interval bounded by \( \alpha_{\text{lo}} \) and \( \alpha_{\text{hi}} \) contains step lengths that satisfy the strong Wolfe conditions;
(b) \( \alpha_{\text{lo}} \) is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value; and
(c) \( \alpha_{\text{hi}} \) is chosen so that \( \phi'(\alpha_{\text{lo}})(\alpha_{\text{hi}} - \alpha_{\text{lo}}) < 0 \).

Each iteration of zoom generates an iterate \( \alpha_j \) between \( \alpha_{\text{lo}} \) and \( \alpha_{\text{hi}} \), and then replaces one of these endpoints by \( \alpha_j \) in such a way that the properties (a), (b), and (c) continue to hold.

**Algorithm 3.6 (zoom).**

repeat
  Interpolate (using quadratic, cubic, or bisection) to find a trial step length \( \alpha_j \) between \( \alpha_{\text{lo}} \) and \( \alpha_{\text{hi}} \);
  Evaluate \( \phi(\alpha_j) \);
  if \( \phi(\alpha_j) > \phi(0) + c_1 \alpha_j \phi'(0) \) or \( \phi(\alpha_j) \geq \phi(\alpha_{\text{lo}}) \)
    \( \alpha_{\text{hi}} \leftarrow \alpha_j \);
  else
    Evaluate \( \phi'(\alpha_j) \);
    if \( |\phi'(\alpha_j)| \leq -c_2 \phi'(0) \)
      Set \( \alpha_s \leftarrow \alpha_j \) and stop;
    if \( \phi'(\alpha_j)(\alpha_{\text{hi}} - \alpha_{\text{lo}}) \geq 0 \)
      \( \alpha_{\text{hi}} \leftarrow \alpha_{\text{lo}} \);
    \( \alpha_{\text{lo}} \leftarrow \alpha_j \);
end (repeat)
If the new estimate \( \alpha_j \) happens to satisfy the strong Wolfe conditions, then zoom has served its purpose of identifying such a point, so it terminates with \( \alpha_s = \alpha_j \). Otherwise, if \( \alpha_j \) satisfies the sufficient decrease condition and has a lower function value than \( x_{i_0} \), then we set \( \alpha_{i_0} \leftarrow \alpha_j \) to maintain condition (b). If this setting results in a violation of condition (c), we remedy the situation by setting \( \alpha_{i_0} \) to the old value of \( \alpha_{i_0} \). Readers should sketch some graphs to see for themselves how zoom works!

As mentioned earlier, the interpolation step that determines \( \alpha_j \) should be safeguarded to ensure that the new step length is not too close to the endpoints of the interval. Practical line search algorithms also make use of the properties of the interpolating polynomials to make educated guesses of where the next step length should lie; see [39, 216]. A problem that can arise is that as the optimization algorithm approaches the solution, two consecutive function values \( f(x_k) \) and \( f(x_{k-1}) \) may be indistinguishable in finite-precision arithmetic. Therefore, the line search must include a stopping test if it cannot attain a lower function value after a certain number (typically, ten) of trial step lengths. Some procedures also stop if the relative change in \( x \) is close to machine precision, or to some user-specified threshold.

A line search algorithm that incorporates all these features is difficult to code. We advocate the use of one of the several good software implementations available in the public domain. See Dennis and Schnabel [92], Lemaréchal [189], Fletcher [101], Moré and Thuente [216] (in particular), and Hager and Zhang [161].

One may ask how much more expensive it is to require the strong Wolfe conditions instead of the regular Wolfe conditions. Our experience suggests that for a “loose” line search (with parameters such as \( c_1 = 10^{-4} \) and \( c_2 = 0.9 \)), both strategies require a similar amount of work. The strong Wolfe conditions have the advantage that by decreasing \( c_2 \) we can directly control the quality of the search, by forcing the accepted value of \( \alpha \) to lie closer to a local minimum. This feature is important in steepest descent or nonlinear conjugate gradient methods, and therefore a step selection routine that enforces the strong Wolfe conditions has wide applicability.

**NOTES AND REFERENCES**

For an extensive discussion of line search termination conditions see Ortega and Rheinboldt [230]. Akaike [2] presents a probabilistic analysis of the steepest descent method with exact line searches on quadratic functions. He shows that when \( n > 2 \), the worst-case bound (3.29) can be expected to hold for most starting points. The case \( n = 2 \) can be studied in closed form; see Bazarra, Sherali, and Shetty [14]. Theorem 3.6 is due to Dennis and Moré.

Some line search methods (see Goldfarb [132] and Moré and Sorensen [213]) compute a direction of negative curvature, whenever it exists, to prevent the iteration from converging to nonminimizing stationary points. A direction of negative curvature \( p_- \) is one that satisfies \( p_- \nabla^2 f(x_k) p_- < 0 \). These algorithms generate a search direction by combining \( p_- \) with the steepest descent direction \( -\nabla f_k \), often performing a curvilinear backtracking line search.
It is difficult to determine the relative contributions of the steepest descent and negative curvature directions. Because of this fact, the approach fell out of favor after the introduction of trust-region methods.

For a more thorough treatment of the modified Cholesky factorization see Gill, Murray, and Wright [130] or Dennis and Schnabel [92]. A modified Cholesky factorization based on Gershgorin disk estimates is described in Schnabel and Eskow [276]. The modified indefinite factorization is from Cheng and Higham [58].

Another strategy for implementing a line search Newton method when the Hessian contains negative eigenvalues is to compute a direction of negative curvature and use it to define the search direction (see Moré and Sorensen [213] and Goldfarb [132]).

Derivative-free line search algorithms include golden section and Fibonacci search. They share some of the features with the line search method given in this chapter. They typically store three trial points that determine an interval containing a one-dimensional minimizer. Golden section and Fibonacci differ in the way in which the trial step lengths are generated; see, for example, [79, 39].

Our discussion of interpolation follows Dennis and Schnabel [92], and the algorithm for finding a step length satisfying the strong Wolfe conditions can be found in Fletcher [101].

---

**Exercises**

1. Program the steepest descent and Newton algorithms using the backtracking line search, Algorithm 3.1. Use them to minimize the Rosenbrock function (2.22). Set the initial step length \( \alpha_0 = 1 \) and print the step length used by each method at each iteration. First try the initial point \( x_0 = (1.2, 1.2)^T \) and then the more difficult starting point \( x_0 = (-1.2, 1)^T \).

2. Show that if \( 0 < c_2 < c_1 < 1 \), there may be no step lengths that satisfy the Wolfe conditions.

3. Show that the one-dimensional minimizer of a strongly convex quadratic function is given by (3.55).

4. Show that the one-dimensional minimizer of a strongly convex quadratic function always satisfies the Goldstein conditions (3.11).

5. Prove that \( \| Bx \| \geq \| x \| / \| B^{-1} \| \) for any nonsingular matrix \( B \). Use this fact to establish (3.19).

6. Consider the steepest descent method with exact line searches applied to the convex quadratic function (3.24). Using the properties given in this chapter, show that if the initial point is such that \( x_0 - x^* \) is parallel to an eigenvector of \( Q \), then the steepest descent method will find the solution in one step.