Chapter 16
DESIGN OF RECURSIVE FILTERS USING OPTIMIZATION METHODS
16.1 Introduction
16.2 Problem Formulation
16.3 Newton's Method

Victoria, BC, Canada
Email: aantoniou@ieee.org

July 14, 2018

Introduction

- Chaps. 11 and 12 have dealt with several methods for the solution of the approximation problem in recursive filters.

Introduction

- Chaps. 11 and 12 have dealt with several methods for the solution of the approximation problem in recursive filters.
- These methods lead to a complete description of the transfer function in closed form, either in terms of its zeros and poles or its coefficients.

Introduction

- Chaps. 11 and 12 have dealt with several methods for the solution of the approximation problem in recursive filters.
- These methods lead to a complete description of the transfer function in closed form, either in terms of its zeros and poles or its coefficients.
- Consequently, they are very efficient and lead to very precise designs.

Introduction

- Chaps. 11 and 12 have dealt with several methods for the solution of the approximation problem in recursive filters.
- These methods lead to a complete description of the transfer function in closed form, either in terms of its zeros and poles or its coefficients.
- Consequently, they are very efficient and lead to very precise designs.
- Their main disadvantage is that they are applicable only for the design of classical-type filters such as Butterworth lowpass filters, elliptic bandpass filters, etc.

Introduction *Cont'd*

- If unusual amplitude or phase responses or delay characteristics are required, then the only available approach for the design of recursive filters is through the use of *optimization methods*.

Introduction *Cont'd*

- If unusual amplitude or phase responses or delay characteristics are required, then the only available approach for the design of recursive filters is through the use of *optimization methods*.

- In these methods, a discrete-time transfer function is assumed and an *error function* is formulated on the basis of some desired amplitude or phase response or some specified group-delay characteristic.

- If unusual amplitude or phase responses or delay characteristics are required, then the only available approach for the design of recursive filters is through the use of *optimization methods*.
- In these methods, a discrete-time transfer function is assumed and an *error function* is formulated on the basis of some desired amplitude or phase response or some specified group-delay characteristic.
- A norm of the error function is then minimized with respect to the transfer-function coefficients.

- If unusual amplitude or phase responses or delay characteristics are required, then the only available approach for the design of recursive filters is through the use of *optimization methods*.

- In these methods, a discrete-time transfer function is assumed and an *error function* is formulated on the basis of some desired amplitude or phase response or some specified group-delay characteristic.

- A norm of the error function is then minimized with respect to the transfer-function coefficients.

- Like the Remez algorithm described in Chap. 15, optimization methods for the design of recursive filters are *iterative*.

  As a result, they usually involve a large amount of computation.

- This presentation is concerned with the application of optimization methods for the design of recursive digital filters.

Introduction *Cont'd*

- This presentation is concerned with the application of optimization methods for the design of recursive digital filters.

  Specific topics to be examined include:

  - formulation of an optimization problem
  - fundamentals of optimization
  - the basic *Newton algorithm*

- This presentation is concerned with the application of optimization methods for the design of recursive digital filters.

  Specific topics to be examined include:

  - formulation of an optimization problem
  - fundamentals of optimization
  - the basic *Newton algorithm*

- More sophisticated practical optimization algorithms such as *quasi-Newton* and *minimax* algorithms along with their application for the design of recursive digital filters will be presented later.

Problem Formulation

The design of a recursive digital filter by optimization involves two general steps:

1. Construct an *objective function* which is proportional on the difference between the actual and specified amplitude or phase response.

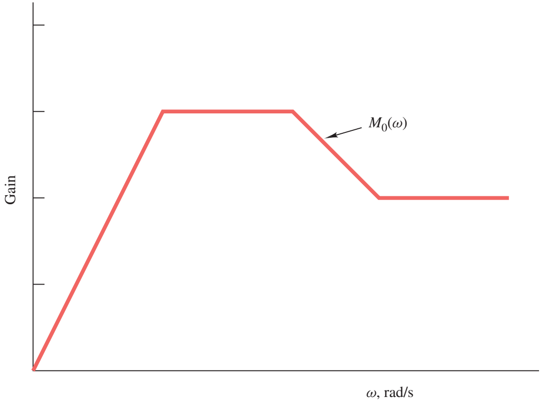
Problem Formulation

The design of a recursive digital filter by optimization involves two general steps:

1. Construct an *objective function* which is proportional on the difference between the actual and specified amplitude or phase response.

2. Minimize the objective function with respect to the transfer function coefficients.

- Let us assume that we need to design an Nth-order recursive filter with a piecewise-linear amplitude response $M_0(\omega)$ such as that shown in the figure.



Problem Formulation *Cont'd*

- The transfer function of the filter can be expressed as

$$H(z) = H_0 \prod_{j=1}^{J} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

where a_{ij} and b_{ij} are real coefficients, $J = N/2$, and H_0 is a positive multiplier constant.

Problem Formulation *Cont'd*

- The transfer function of the filter can be expressed as

$$H(z) = H_0 \prod_{j=1}^{J} \frac{a_{0j} + a_{1j}z + z^2}{b_{0j} + b_{1j}z + z^2}$$

where a_{ij} and b_{ij} are real coefficients, $J = N/2$, and H_0 is a positive multiplier constant.

- As presented, $H(z)$ would be of even order; however, an odd-order $H(z)$ can be obtained by letting

$$a_{0j} = b_{0j} = 0$$

for one value of j.

- The amplitude response of an arbitrary recursive filter can be deduced as

$$M(\mathbf{x}, \omega) = |H(e^{j\omega T})|$$

where ω is the frequency and

$$\mathbf{x} = [a_{01}\ a_{11}\ b_{01}\ b_{11}\ \cdots\ b_{1J}\ H_0]^T$$

is a column vector with $4J + 1$ elements.

Problem Formulation *Cont'd*

- An approximation error can be constructed as the difference between the actual amplitude response $M(\mathbf{x}, \omega)$ and the desired amplitude response $M_0(\omega)$ as

$$e(\mathbf{x}, \omega) = M(\mathbf{x}, \omega) - M_0(\omega)$$

- An approximation error can be constructed as the difference between the actual amplitude response $M(\mathbf{x},\omega)$ and the desired amplitude response $M_0(\omega)$ as

$$e(\mathbf{x},\omega) = M(\mathbf{x},\omega) - M_0(\omega)$$

- By sampling $e(\mathbf{x},\,\omega)$ at frequencies $\omega_1,\ \omega_2,\ \ldots,\ \omega_K$, the column vector

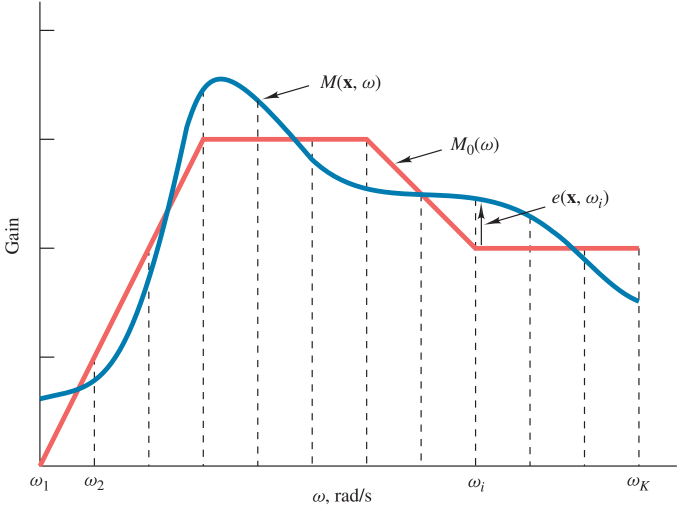
$$\mathbf{E}(\mathbf{x}) = [e_1(\mathbf{x})\ e_2(\mathbf{x})\ \ \ldots\ \ \ e_K(\mathbf{x})]^T$$

can be formed where

$$e_i(\mathbf{x}) = e(\mathbf{x},\,\omega_i)$$

for $i = 1, 2, \ldots, K$.

Problem Formulation *Cont'd*



- A recursive filter can be designed by finding a point $\mathbf{x} = \breve{\mathbf{x}}$ such that

$$e_i(\breve{\mathbf{x}}) \approx 0 \quad \text{for} \quad i = 1, 2, \ldots, K$$

- A recursive filter can be designed by finding a point $\mathbf{x} = \breve{\mathbf{x}}$ such that

$$e_i(\breve{\mathbf{x}}) \approx 0 \quad \text{for} \quad i = 1, 2, \ldots, K$$

- Such a point can be obtained by minimizing the *L_p norm* of $\mathbf{E}(\mathbf{x})$, which is defined as

$$\Psi(\mathbf{x}) = L_p(\mathbf{x}) = ||\mathbf{E}(\mathbf{x})||_p = \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^p\right]^{1/p}$$

where p is a positive integer.

- For filter design, the most important norms are the L_2 and L_∞ norms which are defined as

$$L_2(\mathbf{x}) = \left[\sum_{i=1}^{K} |e_i(\mathbf{x})|^2\right]^{1/2}$$

and
$$\begin{aligned}L_\infty(\mathbf{x}) &= \lim_{p\to\infty}\left\{\sum_{i=1}^{K} |e_i(\mathbf{x})|^p\right\}^{1/p} \\ &= \widehat{E}(\mathbf{x}) \lim_{p\to\infty}\left\{\sum_{i=1}^{K}\left[\frac{|e_i(\mathbf{x})|}{\widehat{E}(\mathbf{x})}\right]^p\right\}^{1/p} \\ &= \widehat{E}(\mathbf{x})\end{aligned}$$

where $\widehat{E}(\mathbf{x}) = \underset{1\leq i\leq K}{max} \; |e_i(\mathbf{x})|$

- In summary, a recursive filter with an amplitude response that approaches a specified amplitude response $M_0(\omega)$ can be designed by solving the optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} \quad \Psi(\mathbf{x})$$

- In summary, a recursive filter with an amplitude response that approaches a specified amplitude response $M_0(\omega)$ can be designed by solving the optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} \;\; \Psi(\mathbf{x})$$

- If

$$\Psi(\mathbf{x}) = L_2(\mathbf{x})$$

a *least-squares* solution is obtained and if

$$\Psi(\mathbf{x}) = L_\infty(\mathbf{x})$$

the outcome will be a so-called *minimax* solution.

Problem Formulation *Cont'd*

- The optimization problem obtained can be solved by using a great variety of *unconstrained* optimization algorithms that have evolved since the invention of computers.

- The optimization problem obtained can be solved by using a great variety of *unconstrained* optimization algorithms that have evolved since the invention of computers.
- In the next series of slides, various implementations of the most fundamental unconstrained optimization algorithm, namely, the *Newton algorithm*, will be presented.

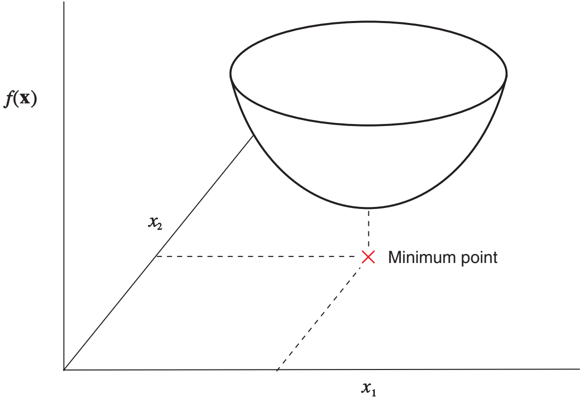
Problem Formulation *Cont'd*

- The optimization problem obtained can be solved by using a great variety of *unconstrained* optimization algorithms that have evolved since the invention of computers.
- In the next series of slides, various implementations of the most fundamental unconstrained optimization algorithm, namely, the *Newton algorithm*, will be presented.
- In due course, the family of *quasi-Newton algorithms* will be described which, as may be expected, are based on the Newton algorithm.

  These algorithms have been found to be *quite robust and very efficient* in many applications including the design of recursive digital filters.

Newton Algorithm

- The Newton algorithm is based on Newton's classical method for finding the minimum of a quadratic *convex* function.



- Consider a function $f(\mathbf{x})$ of n variables, where $\mathbf{x} = [x_1 \; x_2 \; \cdots \; x_n]^T$ is a column vector, and let $\boldsymbol{\delta} = [\delta_1 \; \delta_2 \; \cdots \; \delta_n]^T$ be a change in $\mathbf{x}$.

- Consider a function $f(\mathbf{x})$ of n variables, where $\mathbf{x} = [x_1 \ x_2 \ \cdots \ x_n]^T$ is a column vector, and let $\boldsymbol{\delta} = [\delta_1 \ \delta_2 \ \cdots \ \delta_n]^T$ be a change in $\mathbf{x}$.
- If $f(\mathbf{x})$ has continuous second derivatives, its Taylor series at point $\mathbf{x} + \boldsymbol{\delta}$ is given by

$$f(\mathbf{x}+\boldsymbol{\delta}) = f(\mathbf{x}) + \sum_{i=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_i}\delta_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j}\delta_i\delta_j + o\left(||\boldsymbol{\delta}||_2^2\right)$$

where the remainder $o(||\boldsymbol{\delta}||_2^2)$ approaches zero faster than $||\boldsymbol{\delta}||_2^2$.

- If the remainder is negligible and a *stationary point* exists in the neighborhood of some point $\mathbf{x}$, it can be determined by differentiating $f(\mathbf{x}+\boldsymbol{\delta})$ with respect to elements δ_k for $k = 1, 2, \ldots, n$, and setting the result to zero, i.e.,

$$\frac{\partial f(\mathbf{x}+\boldsymbol{\delta})}{\partial \delta_k} = \frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{i=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_k}\delta_i = 0$$

for $k = 1, 2, \ldots, n$.

- If the remainder is negligible and a *stationary point* exists in the neighborhood of some point $\mathbf{x}$, it can be determined by differentiating $f(\mathbf{x}+\boldsymbol{\delta})$ with respect to elements δ_k for $k = 1, 2, \ldots, n$, and setting the result to zero, i.e.,

$$\frac{\partial f(\mathbf{x}+\boldsymbol{\delta})}{\partial \delta_k} = \frac{\partial f(\mathbf{x})}{\partial x_k} + \sum_{i=1}^{n} \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_k}\delta_i = 0$$

for $k = 1, 2, \ldots, n$.

- The solution of the above equation can be expressed as

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

where $\mathbf{g}$ is the *gradient vector* and $\mathbf{H}$ is the *Hessian matrix*.

...

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

- The gradient and Hessian are given by

$$\mathbf{g} = \nabla f(\mathbf{x}) = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \;\; \frac{\partial f(\mathbf{x})}{\partial x_2} \;\; \cdots \;\; \frac{\partial f(\mathbf{x})}{\partial x_n} \right]^T$$

and

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix}$$

...

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

- This equation will give the solution *if and only if* the following two conditions hold:
  1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
  2. The Hessian is nonsingular.

...

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

- This equation will give the solution *if and only if* the following two conditions hold:
  1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
  2. The Hessian is nonsingular.
- If $f(\mathbf{x})$ is a *quadratic* function, its second partial derivatives are constants, i.e., $\mathbf{H}$ is a constant symmetric matrix, and its third and higher derivatives are zero, i.e., condition (1) holds.

. . .

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

- This equation will give the solution *if and only if* the following two conditions hold:
    1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
    2. The Hessian is nonsingular.
- If $f(\mathbf{x})$ is a *quadratic* function, its second partial derivatives are constants, i.e., $\mathbf{H}$ is a constant symmetric matrix, and its third and higher derivatives are zero, i.e., condition (1) holds.
- If $f(\mathbf{x})$ has a minimum at a stationary point, then the Hessian matrix is *positive definite* at the minimum point.

  In such a case, the Hessian is *nonsingular*, i.e., condition (2) holds.

...

1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
2. The Hessian is nonsingular.

- When the two conditions apply, we go to MATLAB and very quickly we obtain the solution of the optimization problem, i.e., the minimum point and the minimum value of the function, are obtained as

$$\breve{\mathbf{x}} = \mathbf{x} + \breve{\boldsymbol{\delta}}, \quad f(\breve{\mathbf{x}})$$

Newton Algorithm *Cont'd*

. . .

1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
2. The Hessian is nonsingular.

- When the two conditions apply, we go to MATLAB and very quickly we obtain the solution of the optimization problem, i.e., the minimum point and the minimum value of the function, are obtained as

$$\breve{\mathbf{x}} = \mathbf{x} + \breve{\boldsymbol{\delta}}, \quad f(\breve{\mathbf{x}})$$

- No optimization algorithms are necessary.

. . .

1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
2. The Hessian is nonsingular.

- We need optimization algorithms to solve problems that are *not* quadratic.

  In these problems condition (1) and/or condition (2) may be violated.

. . .

1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
2. The Hessian is nonsingular.

- We need optimization algorithms to solve problems that are *not* quadratic.

  In these problems condition (1) and/or condition (2) may be violated.

- If condition (1) is violated, then the equation

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

  will not give the solution.

. . .

1. The remainder $o(||\boldsymbol{\delta}||_2^2)$ can be neglected.
2. The Hessian is nonsingular.

- We need optimization algorithms to solve problems that are *not* quadratic.

  In these problems condition (1) and/or condition (2) may be violated.

- If condition (1) is violated, then the equation

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

  will not give the solution.

- If condition (2) is violated, the equation either has an infinite number of solutions or it has no solutions at all.

- If $f(\mathbf{x})$ is a general *nonquadratic convex function* that has a minimum at point $\breve{\mathbf{x}}$, then in the neighborhood of point $\breve{\mathbf{x}}$ it can be *approximated* by a quadratic function.

- If $f(\mathbf{x})$ is a general *nonquadratic convex function* that has a minimum at point $\breve{\mathbf{x}}$, then in the neighborhood of point $\breve{\mathbf{x}}$ it can be *approximated* by a quadratic function.

  In such a case

  - the remainder $o(||\boldsymbol{\delta}||_2^2)$ becomes negligible, and
  - the second partial derivatives of $f(\mathbf{x})$ become approximately constant.

- If $f(\mathbf{x})$ is a general *nonquadratic convex function* that has a minimum at point $\breve{\mathbf{x}}$, then in the neighborhood of point $\breve{\mathbf{x}}$ it can be *approximated* by a quadratic function.

  In such a case

  - the remainder $o(||\boldsymbol{\delta}||_2^2)$ becomes negligible, and
  - the second partial derivatives of $f(\mathbf{x})$ become approximately constant.

- As a result, in the neighborhood of the solution, conditions (1) and (2) are again satisfied and the equation

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

will yield an accurate estimate of the minimum point.

- Unfortunately, since we do not know the solution, we also do not know where the neighborhood of the solution might be located!

- Unfortunately, since we do not know the solution, we also do not know where the neighborhood of the solution might be located!
- However, if we could find a way of getting near the solution, we would immediately be able to get the solution itself by evaluating

$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$

Newton Algorithm *Cont'd*

- Unfortunately, since we do not know the solution, we also do not know where the neighborhood of the solution might be located!
- However, if we could find a way of getting near the solution, we would immediately be able to get the solution itself by evaluating
$$\boldsymbol{\delta} = -\mathbf{H}^{-1}\mathbf{g}$$
- Thus in order to be able to construct an unconstrained optimization algorithm for the solution of nonquadratic problems, we need to find a mechanism that will enable the algorithm to get to the locale of the solution starting from an arbitrary initial point.

- A basic strategy adopted in most unconstrained optimization algorithms is to apply a series of corrections to an initial point ensuring that each correction is made in a direction that will *reduce* the objective function.

  Such directions are known as *descent directions*.

- A basic strategy adopted in most unconstrained optimization algorithms is to apply a series of corrections to an initial point ensuring that each correction is made in a direction that will *reduce* the objective function.

  Such directions are known as *descent directions*.

- It turns out that if the Hessian $\mathbf{H}$ is positive definite, then so is the inverse Hessian $\mathbf{H}^{-1}$. In such a case, the direction vector

$$\mathbf{d} = -\mathbf{H}^{-1}\mathbf{g}$$

  is a descent direction, as can be easily demonstrated.

- A basic strategy adopted in most unconstrained optimization algorithms is to apply a series of corrections to an initial point ensuring that each correction is made in a direction that will *reduce* the objective function.

  Such directions are known as *descent directions*.

- It turns out that if the Hessian $\mathbf{H}$ is positive definite, then so is the inverse Hessian $\mathbf{H}^{-1}$. In such a case, the direction vector

$$\mathbf{d} = -\mathbf{H}^{-1}\mathbf{g}$$

  is a descent direction, as can be easily demonstrated.

- Since the above descent direction would give the solution in just one shot in a quadratic problem, it is commonly referred to as the *Newton direction*.

- If the problem is nonquadratic, the Newton direction will not give the solution.

  Nevertheless, it will reduce the value of the objective function by a certain amount and, therefore, a new point will be obtained that is closer to the solution.

- If the problem is nonquadratic, the Newton direction will not give the solution.

  Nevertheless, it will reduce the value of the objective function by a certain amount and, therefore, a new point will be obtained that is closer to the solution.

- To maximize the reduction achieved in the objective function and thereby get closer to the solution, we let

$$\boldsymbol{\delta} = \alpha \mathbf{d}$$

  and choose parameter α such that the objective function, $f(\boldsymbol{x} + \alpha \mathbf{d})$, is minimized.

  We can do that by using a 1-dimensional optimization algorithm also known as a *line search*.

- Most of the elements of an unconstrained optimization algorithm are now in place except that in a nonquadratic problem the Hessian *may sometimes become nonpositive definite*.

  In such a case, the Newton direction would become an *ascent direction*!

- Most of the elements of an unconstrained optimization algorithm are now in place except that in a nonquadratic problem the Hessian *may sometimes become nonpositive definite*.

  In such a case, the Newton direction would become an *ascent direction*!

- To circumvent this problem, we simply *force* the Hessian to become positive definite, for example, we could assign

$$\mathbf{H} = \mathbf{I}$$

  where $\mathbf{I}$ is the identity matrix.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.
3. Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.

Newton Algorithm *Cont'd*

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.
3. Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.
4. Find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\,\mathbf{d}_k)$, using a line search.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.
3. Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.
4. Find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\,\mathbf{d}_k)$, using a line search.
5. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k$, where $\boldsymbol{\delta}_k = \alpha_k\mathbf{d}_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.
3. Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.
4. Find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\,\mathbf{d}_k)$, using a line search.
5. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k$, where $\boldsymbol{\delta}_k = \alpha_k\mathbf{d}_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
6. If $||\alpha_k\mathbf{d}_k||_2 < \varepsilon$, then output $\breve{\mathbf{x}} = \mathbf{x}_{k+1}$, $f(\breve{\mathbf{x}}) = f_{k+1}$, and stop.

1. Input $\mathbf{x}_0$, ε, and set $k = 0$.
2. Compute the gradient $\mathbf{g}_k$ and Hessian $\mathbf{H}_k$.

   If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.
3. Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.
4. Find α_k, the value of α that minimizes $f(\mathbf{x}_k + \alpha\,\mathbf{d}_k)$, using a line search.
5. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k$, where $\boldsymbol{\delta}_k = \alpha_k\mathbf{d}_k$, and compute $f_{k+1} = f(\mathbf{x}_{k+1})$.
6. If $||\alpha_k\mathbf{d}_k||_2 < \varepsilon$, then output $\breve{\mathbf{x}} = \mathbf{x}_{k+1}$, $f(\breve{\mathbf{x}}) = f_{k+1}$, and stop.

   Otherwise, set $k = k + 1$ and repeat from step 2.

Newton Algorithm *Cont'd*

- In Step 4 of the algorithm, a *line search* is used to find the value of α_k that minimizes the value of the objective function, $f(\mathbf{x}_k + \alpha\,\mathbf{d}_k)$, along the Newton direction.

- In Step 4 of the algorithm, a *line search* is used to find the value of α_k that minimizes the value of the objective function, $f(\mathbf{x}_k + \alpha\, \mathbf{d}_k)$, along the Newton direction.
- Any *1-dimensional optimization algorithm* can be used as a line search.Therefore, a great variety of possible line searches

  are available. See Chap. 4 of

  A. Antoniou and W.-S. Lu, *Practical Optimization: Algorithms and Engineering Applications*, Springer, 2007, at the following link:

  http://www.ece.uvic.ca/~optimize

- In Step 6, the algorithm is terminated if the L_2 norm of $\alpha_k \mathbf{d}_k$, i.e., the *magnitude of the change in* $\mathbf{x}$, is less than ε.

  The parameter ε is said to be the *termination tolerance* and it is a small positive constant whose value is determined by the application under consideration.

- In Step 6, the algorithm is terminated if the L_2 norm of $\alpha_k \mathbf{d}_k$, i.e., the *magnitude of the change in* $\mathbf{x}$, is less than ε.

  The parameter ε is said to be the *termination tolerance* and it is a small positive constant whose value is determined by the application under consideration.

- In certain applications, a termination tolerance on the *change in the objective function* itself, e.g., $|f_{k+1} - f_k| < \varepsilon$, may be preferable and sometimes termination tolerances may be imposed on the magnitudes of *both* the changes in $\mathbf{x}$ and the objective function.

- Considering a nonquadratic convex problem, starting from an arbitrary initial point, the Newton algorithm computes a series of corrections to the initial point, in each case minimizing the objective function along a Newton direction.

Newton Algorithm *Cont'd*

- Considering a nonquadratic convex problem, starting from an arbitrary initial point, the Newton algorithm computes a series of corrections to the initial point, in each case minimizing the objective function along a Newton direction.
- In this way, new points are generated that are getting closer and closer to the solution.

- Considering a nonquadratic convex problem, starting from an arbitrary initial point, the Newton algorithm computes a series of corrections to the initial point, in each case minimizing the objective function along a Newton direction.
- In this way, new points are generated that are getting closer and closer to the solution.
- As the solution is approached, the objective function behaves more and more like a quadratic function, parameter α assumes values that are closer and closer to unity, and the Newton directions give new points that are closer and closer to the solution.

Newton Algorithm *Cont'd*

- Considering a nonquadratic convex problem, starting from an arbitrary initial point, the Newton algorithm computes a series of corrections to the initial point, in each case minimizing the objective function along a Newton direction.
- In this way, new points are generated that are getting closer and closer to the solution.
- As the solution is approached, the objective function behaves more and more like a quadratic function, parameter α assumes values that are closer and closer to unity, and the Newton directions give new points that are closer and closer to the solution.
- Eventually, a point is obtained that is sufficiently close to the solution and convergence is deemed to have been achieved.

- A characteristic feature of the basic Newton algorithm is that at the beginning the algorithm makes small but steady reductions in the objective function from one iteration to the next.

Newton Algorithm *Cont'd*

- A characteristic feature of the basic Newton algorithm is that at the beginning the algorithm makes small but steady reductions in the objective function from one iteration to the next.
- As the solution is approached, progress becomes very rapid and the algorithm appears to *zoom* to the solution.

- A characteristic feature of the basic Newton algorithm is that at the beginning the algorithm makes small but steady reductions in the objective function from one iteration to the next.
- As the solution is approached, progress becomes very rapid and the algorithm appears to *zoom* to the solution.
- This feature is shared by all the algorithms based on Newton's classical method for finding the minima of a function, including the family of quasi-Newton algorithms.

- So far, we have assumed that we are dealing with a *convex* optimization problem that has a unique *global* minimum.

- So far, we have assumed that we are dealing with a *convex* optimization problem that has a unique *global* minimum.
- In practice, the problem may have more than one local minimum, sometimes a large number, and on occasion a well-defined minimum may not even exist.

- So far, we have assumed that we are dealing with a *convex* optimization problem that has a unique *global* minimum.
- In practice, the problem may have more than one local minimum, sometimes a large number, and on occasion a well-defined minimum may not even exist.
- We must, therefore, abandon the expectation that we shall always be able to obtain the best solution available.

- So far, we have assumed that we are dealing with a *convex* optimization problem that has a unique *global* minimum.
- In practice, the problem may have more than one local minimum, sometimes a large number, and on occasion a well-defined minimum may not even exist.
- We must, therefore, abandon the expectation that we shall always be able to obtain the best solution available.
- The best we can hope for is a solution that satisfies the required specifications.

- So far, we have assumed that we are dealing with a *convex* optimization problem that has a unique *global* minimum.
- In practice, the problem may have more than one local minimum, sometimes a large number, and on occasion a well-defined minimum may not even exist.
- We must, therefore, abandon the expectation that we shall always be able to obtain the best solution available.
- The best we can hope for is a solution that satisfies the required specifications.
- This is usually achieved by using different initialization points.

This slide concludes the presentation.

Thank you for your attention.