Unconstrained optimisation

(CK0031/CK0248)

Francesco Corona

Department of Computer Science
Federal University of Ceará, Fortaleza

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation

Minimisation (maximisation)

$\rightsquigarrow$ Find a global or local minimum (maximum) of some objective function

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation (cont.)

Definition

*Let $f : \mathcal{R}^n \rightarrow \mathcal{R}$ with $n \geq 1$ be a **cost** or an **objective function***

*The **unconstrained optimisation** problem*

$$\rightsquigarrow \quad \min_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x}) \tag{1}$$

*The **constrained optimisation** problem*

$$\rightsquigarrow \quad \min_{\mathbf{x} \in \Omega \subset \mathcal{R}^n} f(\mathbf{x}) \tag{2}$$

$\Omega \subset \mathcal{R}^n$ is a closed subset determined by equality and inequality constraints

- *Constraints are dictated by the nature of the problem to solve*

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation (cont.)

Example

Find the optimal allocation of $i = 1, \dots, n$ bounded resources x_i

$\rightsquigarrow$ Bounded resources means limited resources

The constraints express these limits in terms of inequalities

$$0 \leq x_i \leq C_i, \quad \text{with } C_i \text{ some given constants}$$

The set $\Omega = \big\{\mathbf{x} = (x_1, \dots, x_n) : 0 \leq x_i \leq C_i, i = 1, \dots, n\big\}$

- A subset of $\mathcal{R}^n$ determined by such constraints

■

Numerical optimisation (cont.)

For some problems, Ω is characterised by explicit conditions

$\rightsquigarrow$ **Equality constraints**

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

$\rightsquigarrow$ **Inequality constraints**

$$\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$$

$\mathbf{h} : \mathcal{R}^n \rightarrow \mathcal{R}^m$ with $m \leq n$ indicates some function of $\mathbf{x}$

- $\mathbf{h}(\mathbf{x}) \leq \mathbf{0}$ corresponds to $h_i(x_1, x_2, \cdots, x_n) \leq 0$, for $i = 1, \ldots, m$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation (cont.)

Definition

Let f be a continuous function and let Ω be a connected set

A constrained optimisation problem is a non-linear programming problem

Convex programming

$\rightsquigarrow$ *f is a convex function and $\mathbf{h}$ has convex components*

Linear programming

$\rightsquigarrow$ *f and $\mathbf{h}$ are linear*

Quadratic programming

$\rightsquigarrow$ *f is quadratic and $\mathbf{h}$ is linear*

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation (cont.)

Remark

Minimisation v maximisation

Computing the maximum of function f is equivalent to computing the minimum of function $g = -f$

$\rightsquigarrow$ We shall only consider minimisation algorithms

Definition

The minimum value of some given objective function is interesting

The point at which such minimum is achieved is more interesting

$\rightsquigarrow$ *Such point is called minimiser*

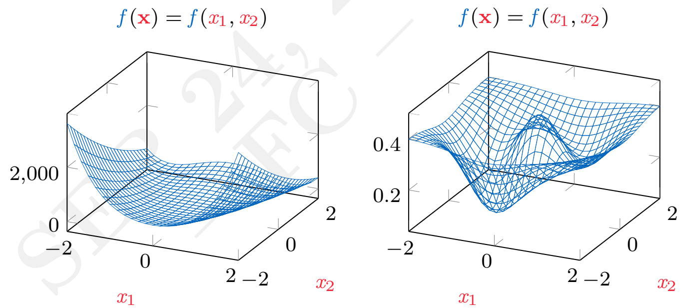
Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Numerical optimisation (cont.)

We consider the numerical solutions of optimisation problems

The ideal situation is a function with *unique global* minimiser

- There are often several (local) minimisers



Numerical optimisation (cont.)

We are interested in finding either a (good) local or the global minimiser

- This is the simple meaning of minimising an objective function

Definition

*Point $\mathbf{x}^*$ is a **global minimiser** of f*

$\rightsquigarrow$ *if $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{R}^n$*

*Point $\mathbf{x}^*$ is a **local minimiser** of f*

$\rightsquigarrow$ *if there is a $B_r(\mathbf{x}^*) \subset \mathcal{R}^n$, a ball centred in $\mathbf{x}^*$ and radius $r > 0$, such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in B_r(\mathbf{x}^*)$*

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Definition

Let f be differentiable in $\mathcal{R}^n$ with first and second derivatives

*Let **gradient vector** of f at point $\mathbf{x} \in \mathcal{R}^n$ be the symbol*

$$\nabla f(\mathbf{x}) = \Big[\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n}\Big]^T \tag{3}$$

*Let **Hessian matrix** of f at point $\mathbf{x} \in \mathcal{R}^n$ be the symbol*

$$\mathbf{H}(\mathbf{x}) = (h_{ij})_{i,j=1}^n, \quad \text{with } h_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i} \tag{4}$$

In general, it will be assumed that problem functions are smooth

- Continuous and continuously (Frétchet) differentiable, $\mathcal{C}^1$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

For $f(\mathbf{x})$ at any point $\mathbf{x}$ there is a **vector of first derivatives**

$\rightsquigarrow$ **Gradient vector**

$$\begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix}_{\mathbf{x}} = \nabla f(\mathbf{x}) \tag{5}$$

∇ is the gradient operator $\Big[\partial/\partial x_1, \partial/\partial x_2, \cdots, \partial/\partial x_n\Big]^T$

Unconstrained optimisation (cont.)

Let $f(\mathbf{x})$ be twice-differentiable, $\mathcal{C}^2$

There is a **matrix of second partial derivatives**

$\rightsquigarrow$ **Hessian matrix**

$$\begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}_{\mathbf{x}} = \mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}) \qquad (6)$$

The (i,j)-th element of the Hessian matrix, $\partial^2 f/(\partial x_i \partial x_j)$

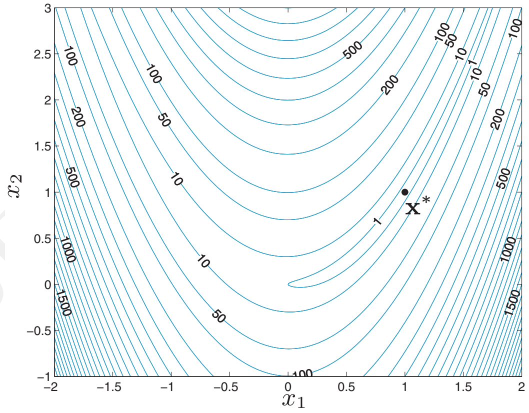
Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Example

Rosenbrock's function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



The global minimum is at $\mathbf{x}^* = (1,1)$, and variation around $\mathbf{x}^*$ is low

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

A test-function for optimisation methods

$$f(\mathbf{x}) = f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- $\partial f/\partial x_1 = -400 x_1 (x_2 - x_1^2) - 2(1 - x_1)$
- $\partial f/\partial x_2 = 200(x_2 - x_1^2)$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \end{bmatrix}_{\mathbf{x}} = \begin{bmatrix} -400 x_1 (x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}_{\mathbf{x}} \qquad (7)$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

$$f(\mathbf{x}) = f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

- $\partial f/\partial x_1 = -400 x_1 (x_2 - x_1^2) - 2(1 - x_1)$
- $\partial f/\partial x_2 = 200(x_2 - x_1^2)$

- $\partial^2 f/(\partial x_1 \partial x_1) = 1200 x_1^2 - 400 x_2 + 2$
- $\partial^2 f/(\partial x_1 \partial x_2) = -400 x_1$
- $\partial^2 f/(\partial x_2 \partial x_1) = -400 x_1$
- $\partial^2 f/(\partial x_2 \partial x_2) = 200$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 x_2} \end{bmatrix}_{\mathbf{x}} = \begin{bmatrix} 1200 x_1^2 - 400 x_2 + 2 & -400 x_1 \\ -400 x_1 & 200 \end{bmatrix}_{\mathbf{x}} \qquad (8)$$

Unconstrained optimisation (cont.)

In general, ∇f and $\nabla^2 f$ will vary with $\mathbf{x}$

At $\mathbf{x}' = (0,0)^T$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}_{\mathbf{x}} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}_{\mathbf{x}=(0,0)^T}$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}_{\mathbf{x}} = \begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}_{\mathbf{x}=(0,0)^T}$$

■

Unconstrained optimisation (cont.)

The idea of a line is also important

Definition

We can define the line as a set of points

$$\mathbf{x}\big[\ = [\mathbf{x}(\alpha)] = \mathbf{x}' + \alpha\mathbf{d}, \quad \textit{for all } \alpha$$

$\mathbf{x}'$ *is some fixed point along the line*

- *It corresponds to* $\alpha = 0$

$\mathbf{d}$ *is the direction of the line*

Unconstrained optimisation (cont.)

Example

Let the fixed point $\mathbf{x}'$ be the point $(2,2)$, let the direction $\mathbf{d}$ be $(3,1)$

⇝ Draw the line $\mathbf{x} = \mathbf{x}' + \alpha\mathbf{d}$, for all α

■

Unconstrained optimisation (cont.)

We can determine expressions for the derivatives of f along any line $\mathbf{x}(\alpha)$

⇝ Based on definitions of line, gradient vector and Hessian matrix

By the chain rule of derivation of scalar-valued function of a vector

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\big\{\cdot\,[\mathbf{x}(\alpha)]\big\} = \sum_i \frac{\mathrm{d}x_i(\alpha)}{\mathrm{d}\alpha}\frac{\partial}{\partial x_i}\big\{\cdot\,[\mathbf{x}(\alpha)]\big\} = \sum_i d_i \frac{\partial}{\partial x_i}\big\{\cdot\,[\mathbf{x}(\alpha)]\big\} = \mathbf{d}^T\nabla\big\{\cdot\,[\mathbf{x}(\alpha)]\big\}$$

The slope of $f\{= f[\mathbf{x}(\alpha)]\}$ along the line at any point $\mathbf{x}(\alpha)$

$$\frac{\mathrm{d}f}{\mathrm{d}\alpha} = \mathbf{d}^T\nabla f = \nabla f^T\mathbf{d}$$

This is the directional derivative of f with respect to $\mathbf{d}$

- ∇f is calculated at $\mathbf{x}(\alpha)$

Unconstrained optimisation (cont.)

The curvature of $f\{= f[\mathbf{x}(\alpha)]\}$ along the line at any point $\mathbf{x}(\alpha)$

$$\frac{\mathrm{d}^2 f}{\mathrm{d}\alpha^2} = \frac{\mathrm{d}}{\mathrm{d}\alpha}\Big(\frac{\mathrm{d}f}{\mathrm{d}\alpha}\Big) = \mathbf{d}^T\nabla(\nabla f^T\mathbf{d}) = \mathbf{d}^T\nabla^2 f\mathbf{d}$$

This is the second-order directional derivative of f

- ∇f and $\nabla^2 f$ are calculated at $\mathbf{x}(\alpha)$

Let $\mathbf{G} = \nabla^2 f$, then $\mathbf{Gd}$ is a vector

$$(\mathbf{Gd})_i = \sum_j G_{ij} d_j$$

$\mathbf{d}^T\mathbf{Gd}$ is the scalar product of $\mathbf{d}$ and $\mathbf{Gd}$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Example

Rosenbrock's function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}_{\mathbf{x}=\mathbf{0}}$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}_{\mathbf{x}}$$

We consider point $\mathbf{x}' = (0, 0)^T$

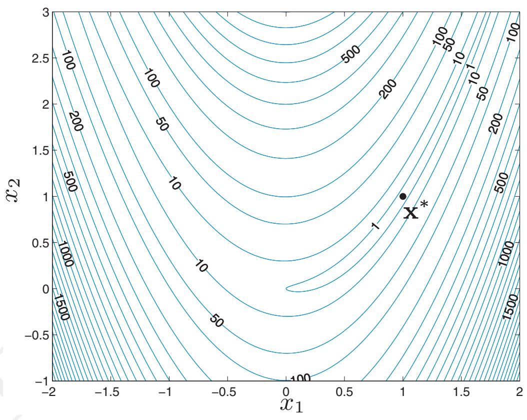
$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}_{\mathbf{x}}$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}_{\mathbf{x}}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

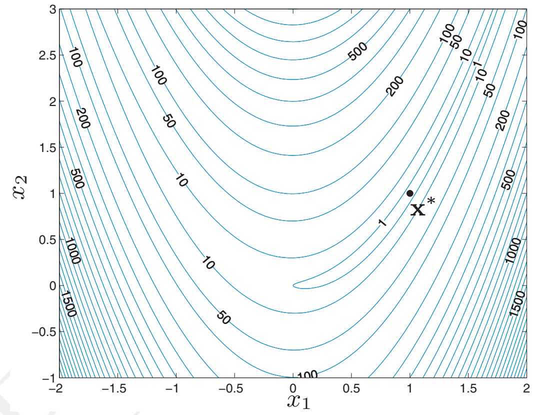


The slope at point $\mathbf{x}' = (0, 0)^T$, along the line with direction $\mathbf{d} = (1, 0)^T$

$$\mathbf{d}^T\nabla f(\mathbf{x}') = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} -2 \\ 0 \end{bmatrix} = -2$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)



The curvature at point $\mathbf{x}' = (0, 0)^T$, along the line with direction $\mathbf{d} = (1, 0)^T$

$$\mathbf{d}^T\mathbf{Gd} = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}}_{\begin{bmatrix} 2 & 0 \end{bmatrix}^T}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = 2$$

Unconstrained optimisation (cont.)

Definition

Let $f \in \mathcal{C}^2(\mathcal{R}^n)$, all first and second derivatives exist and are continuous

Then, $\mathbf{H}(\mathbf{x})$ is symmetric for every $\mathbf{x} \in \mathcal{R}^n$

Definition

*A point $\mathbf{x}^*$ is called a **stationary** or **critical point** for f if $\nabla f(\mathbf{x}^*) = \mathbf{0}$*

*A point such that $\nabla f(\mathbf{x}^*) \neq \mathbf{0}$ is called a **regular point***

A function f over $\mathcal{R}^n$ does not necessarily admit a minimiser

- Should this point exist it is not necessarily unique

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Example

- $f(\mathbf{x}) = x_1 + 3x_2$ is unbounded in $\mathcal{R}^2$
- $f(\mathbf{x}) = \sin(x_1)\sin(x_2)\cdots\sin(x_n)$ admits an infinite number of minimisers and maximisers in $\mathcal{R}^n$, they are both local and global

■

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Definition

*Function $f : \Omega \subseteq \mathcal{R}^n \to \mathcal{R}$ is **convex** in Ω if*

$$\rightsquigarrow \quad f\big[\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\big] \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}), \quad \forall \mathbf{x}, \mathbf{y} \in \Omega \qquad (9)$$

for all $\alpha \in [0, 1]$

Definition

*Function f is **Lipschitz** in Ω if*

$$\rightsquigarrow \quad ||f(\mathbf{x}) - f(\mathbf{y})|| \leq L||\mathbf{x} - \mathbf{y}||, \quad \forall \mathbf{x}, \mathbf{y} \in \Omega \qquad (10)$$

for some constant $L > 0$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Proposition

Optimality conditions

Let $\mathbf{x}^* \in \mathcal{R}^n$ and $r > 0$ exists such that $f \in \mathcal{C}^1[B_r(\mathbf{x}^*)]$

- *If $\mathbf{x}^*$ is a minimiser for f (local or global), then $\nabla f(\mathbf{x}^*) = \mathbf{0}$*
  - *Also, if $f \in \mathcal{C}^2[B_r(\mathbf{x}^*)]$, $\mathbf{H}(\mathbf{x}^*)$ is positive semidefinite (PSD)*

Let $\mathbf{x}^* \in \mathcal{R}^n$ and $r > 0$ exists such that $f \in \mathcal{C}^2[B_r(\mathbf{x}^*)]$

- *If $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{H}(\mathbf{x}^*)$ is positive definite (PD) for all $\mathbf{x} \in B_r(\mathbf{x}^*)$, then $\mathbf{x}^*$ is a local minimiser of f*
- *If $f \in \mathcal{C}^1(\mathcal{R}^n)$ is convex in $\mathcal{R}^n$ and $\nabla f(\mathbf{x}^*) = \mathbf{0}$, then $\mathbf{x}^*$ is a global minimiser for f*

Unconstrained optimisation (cont.)

Definition

A symmetric real matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$ is ***positive definite (PD)*** *if*

$$\forall \mathbf{x} \in \mathcal{R}^n \text{ with } \mathbf{x} \neq \mathbf{0}, \quad \rightsquigarrow \quad \mathbf{x}^T \mathbf{A} \mathbf{x} > 0$$

A symmetric real matrix $\mathbf{A} \in \mathcal{R}^{n \times n}$ is ***positive semidefinite (PSD)*** *if*

$$\forall \mathbf{x} \in \mathcal{R}^n \text{ with } \mathbf{x} \neq \mathbf{0}, \quad \rightsquigarrow \quad \mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$$

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Most methods for numerical optimisation are of iterative type

- They can be classified into two main categories

Depending on whether they use derivatives of the objective

Derivative-free methods

- ⇝ They explore the local behaviour of a cost function
- ⇝ Direct comparison between function values

Methods using derivatives

- ⇝ They use local information of the objective

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Unconstrained optimisation (cont.)

Methods based on derivatives are expected faster convergence

Remark

It can be shown that given $\overline{\mathbf{x}} \in \text{dom}(f)$, if $\nabla f(\overline{\mathbf{x}})$ exists and it is not null, then the largest increase of f from $\overline{\mathbf{x}}$ is along the gradient vector

Conversely, the largest decrease is along the opposite direction

Among them, the two most important classes of techniques

- ⇝ **Trust-region methods**
- ⇝ **Line-search methods**

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method

Numerical optimisation

The Newton method

Let $f : \mathcal{R}^n \rightarrow \mathcal{R}$ with $n \geq 1$ be of class $\mathcal{C}^2(\mathcal{R}^n)$

We know how to compute its first and second order partial derivatives

We apply Newton's method to solve a system of nonlinear equation

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

Remark

Newton's method

Consider the problem of finding the zero of some $f : [a, b] \subset \mathcal{R} \rightarrow \mathcal{R}$

$$\rightsquigarrow \quad \text{Find } \alpha \in [a, b] \text{ such that } f(\alpha) = 0$$

We know the equation of the tangent to function $f(x)$ at some point $x^{(k)}$

$$y(x) = f\big[x^{(k)}\big] + f'\big[x^{(k)}\big]\big[x - x^{(k)}\big]$$

We can solve for some point $x = x^{(k+1)}$, such that $y\big[x^{(k+1)}\big] = 0$

$$\rightsquigarrow \quad x^{(k+1)} = x^{(k)} - \frac{f\big[x^{(k)}\big]}{f'\big[x^{(k)}\big]}$$

All this, for $k = 0, 1, 2, \dots$ and $f'\big[x^{(k)}\big] \neq 0$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

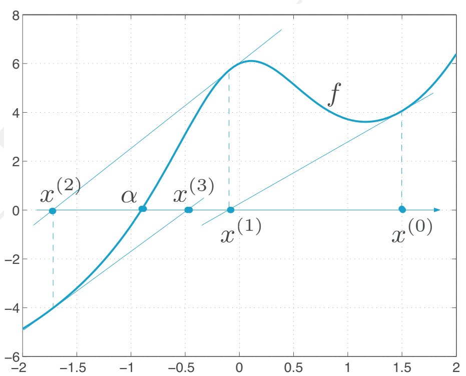
The Newton method (cont.)

$$y(x) = f\big[x^{(k)}\big] + f'\big[x^{(k)}\big]\big[x - x^{(k)}\big]$$

$$x^{(k+1)} = x^{(k)} - \frac{f\big[x^{(k)}\big]}{f'\big[x^{(k)}\big]}$$

Sequence $\{x^{(k)}\}$ is the **Newton method** for finding the zero of function f

$\rightsquigarrow$ The method reduces to locally substituting f with its tangent



Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

Consider now a set of nonlinear equations

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ &\cdots \\ f_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned}$$

We re-write the system in vector form

- Let $\mathbf{f} \equiv (f_1, \dots, f_n)^T$
- Let $\mathbf{x} \equiv (x_1, \dots, x_n)^T$

$$\rightsquigarrow \quad \mathbf{f}(\mathbf{x}) = \mathbf{0}$$

We want solve the system of nonlinear equation

$\rightsquigarrow$ We can extend Newton's method

Replace first derivative of function f with Jacobian $\mathbf{J_f}$ of function $\mathbf{f}$

$$\rightsquigarrow \quad (\mathbf{J_f})_{ij} \equiv \frac{\partial f_i}{\partial x_j}, \quad \text{with } i, j = 1, \dots, n$$

The Newton method (cont.)

Consider the general system of nonlinear equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$

$$f_1(x_1, \dots, x_j, \dots, x_n) = 0$$
$$\cdots$$
$$f_i(x_1, \dots, x_j, \dots, x_n) = 0$$
$$\cdots$$
$$f_n(x_1, \dots, x_j, \dots, x_n) = 0$$

The corresponding Jacobian matrix

$$\mathbf{J_f}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

The Newton method (cont.)

Given this notation, we define the multivariable Newton's method[1]

Pseudo-code

Let $\mathbf{x}^{(0)} \in \mathcal{R}^n$ *be an initial solution*

For $k = 0, 1, \dots,$ *until convergence*

Solve $\mathbf{J_f}\big[\mathbf{x}^{(k)}\big]\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{f}\big[\mathbf{x}^{(k)}\big]$

Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$

At each iteration, a linear system with matrix $\mathbf{J_f}\big[\mathbf{x}^{(k)}\big]$ must be solved

[1] $x^{(k+1)} = x^{(k)} - f\big[x^{(k)}\big]/f'\big[x^{(k)}\big]$, $\delta_x^{(k)} = x^{(k+1)} - x^{(k)}$, $\rightsquigarrow f'[x^{(k)}]\delta_x^{(k)} = -f[x^{(k)}]$

The Newton method (cont.)

```
function [x,res,iter] = sNWT(F_fun,J_fun,x_0,tol,imx)
% [ROOT,RES,ITER]=SNWT(F_FUN,J_FUN,X_0,TOL,IMX) Calculate vector ROOT,
% the zero of a nonlinear system defined in F_FUN with Jacobian J_FUN,
% from initial point X_0
%
% RES is residual in ROOT and ITER is number of iterations
% F_FUN e J_FUN are external functions (as M-files)

iter=0; x=x_0; err=1+tol;

while err >= tol & iter < imx
 J = J_fun(x);
 F = F_fun(x);
 deltax = -J\F;                       %(Matlab/Octave backslash operator)
 x = x + deltax;
 err = norm(deltax); iter = 1+iter;
end
res = norm(F_fun(x));

if(iter==imx & err > tol)
 disp('[Out by KMAX]');
else
 disp('[Out by TOL]'));
end
return
```


The Newton method (cont.)

```
function J = J_fun(x)
 J(1,1) = dF_1 / dx_1;                    % Add your own expression
 J(1,2) = dF_1 / dx_2;                    % Add your own expression
 ...

 J(2,1) = dF_2 / dx_1;                    % Add your own expression
 J(2,2) = dF_2 / dx_2;                    % Add your own expression
 ...

 J(N,1) = dF_N / dx_1;                    % Add your own expression
 J(N,2) = dF_N / dx_2;                    % Add your own expression
 ...

 return
```

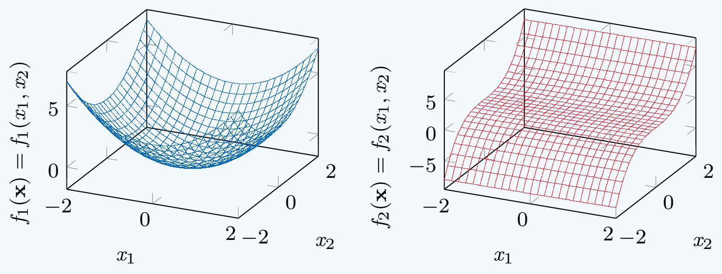
```
function F = F_fun(x)
 F(1,1) = F_1(x_1,x_2,...);               % Add your own expression
 F(2,1) = F_2(x_1,x_2,...);               % Add your own expression
 ...
 F(N,1) = F_N(x_1,x_2,...);               % Add your own expression

return
```

The Newton method (cont.)

Example

Consider the nonlinear system of equations

$$\begin{cases} f_1(x_1, x_2) = x_1^2 + x_2^2 = 1 \\ f_2(x_1, x_2) = \sin\left(\dfrac{\pi}{2} x_1\right) + x_2^3 = 0 \end{cases}$$



The system has two solutions

- $\approx (0.47, -0.88)$ and $\approx (-0.47, 0.88)$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

$$\begin{cases} f_1(x_1, x_2) = x_1^2 + x_2^2 = 1 \\ f_2(x_1, x_2) = \sin\left(\dfrac{\pi}{2} x_1\right) + x_2^3 = 0 \end{cases}$$

```
function F=F_fun(x)
 hpi = 0.5*pi;
 F(1,1) = x(1)^2 + x(2)^2 = 1;
 F(2,1) = sin(pih*x(1)) + x(2)^3 = 0;
return
```

```
function J=J_fun(x)
 hpi = 0.5*pi;
 J(1,1) = 2*x(1);
 J(1,2) = 2*x(2);
 J(2,1) = hpi*cos(hpi*x(1));
 J(2,2) = 3*x(2)^2;
return
```

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

Suppose we start the solution from point $\mathbf{x}^{(0)} = (1, 1)^T$

Let $\varepsilon = 0.00001$ be the user-defined tolerance

```
x_0=[1;1]; % Initial solution
tol=1e-5;  % Tolerance
imx=20;    % Iteration

[x,res,iter] = sNWT(@F_fun,@J_fun,x_0,tol,imx);
```

■

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

$$\mathbf{f}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{0}$$

⇝ The Jacobian $\mathbf{J_f}[\mathbf{x}^{(k)}]$ of the system is the Hessian matrix $\mathbf{H}(\mathbf{x})$ of f

⇝ As computed at the generic iteration point $\mathbf{x}^{(k)}$

Pseudo-code

Given $\mathbf{x}^{(0)} \in \mathcal{R}^n$, *for* $k = 0, 1, \dots,$ *until convergence*

$$\text{Solve} \quad \underbrace{\mathbf{H}[\mathbf{x}^{(k)}]}_{\mathbf{J_f}[\mathbf{x}^{(k)}]} \delta\mathbf{x}^{(k)} = -\underbrace{\nabla f[\mathbf{x}^{(k)}]}_{\mathbf{f}[\mathbf{x}^{(k)}]} \qquad \text{Set} \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)} \tag{11}$$

We can define a suitable *stopping test/criterion*

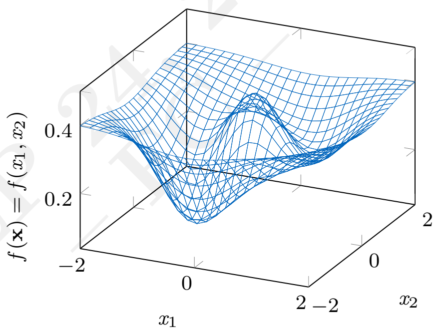
$$||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| \leq \varepsilon$$

$\varepsilon > 0$ is an appropriate *tolerance value*

The Newton method (cont.)

Example

$$f(\mathbf{x}) = 2/5 - 1/10(5x_1^2 + 5x_2^2 + 3x_1x_2 - x_1 - 2x_2)e^{\left[-\left(x_1^2+x_2^2\right)\right]}$$



We want to approximate the global minimum $\mathbf{x}^* \approx (-0.63, -0.70)$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

Netwon's method with a tolerance $\varepsilon = 10^{-5}$

Let $\mathbf{x}^{(0)} = (-0.9, -0.9)$

⇝ After 5 iterations the method converges to `x=[-0.63058;-0.70074]`

Let $\mathbf{x}^{(0)} = (-1.0, -1.0)$

⇝ After 400 iterations the stopping criterion is still not fulfilled

Let $\mathbf{x}^{(0)} = (+0.5, -0.5)$

⇝ After 5 iterations the method converges to the saddle point

- `x=[0.80659; -0.54010]`

Note that Newton's method may converge to any stationary point

- (A point that is not necessarily a minimiser)

■

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

Remark

A necessary condition for convergence of Newton's method

- $\mathbf{x}^{(0)}$ should be sufficiently close to the minimiser $\mathbf{x}^*$

The **local convergence property** of the method

Remark

General convergence criterium for the Newton's method

If $f \in \mathcal{C}^2(\mathcal{R}^n)$ with stationary point $\mathbf{x}^*$

⇝ Positive definite Hessian $\mathbf{H}(\mathbf{x}^*)$

⇝ Lipschitz continuous components of $\mathbf{H}(\mathbf{x})$ in a neighbourhood of $\mathbf{x}^*$

Then, for $\mathbf{x}^{(0)}$ sufficiently close to $\mathbf{x}^*$, it converges (quadratically) to $\mathbf{x}^*$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Newton method (cont.)

In spite of a simple implementation, the method is demanding for large n

⇝ It requires the analytic expression of the derivatives

⇝ The computation of both gradient and Hessian of f

- (Gradient and Hessian at each iteration)

Let alone that $\mathbf{x}^{(0)}$ has to be chosen near enough $\mathbf{x}^*$

Remark

A valid approach to design efficient and robust minimisation algorithms

⇝ Combine locally- with globally-convergent methods

Global convergence guarantees convergence to a stationary point

- (Not necessarily a global minimiser), for all $\mathbf{x}^{(0)} \in \mathcal{R}^n$

Line-search methods
Numerical optimisation

Line-search methods

Line-search or **descent methods** are iterative methods

We suppose that the objective function $f \in \mathcal{C}^2(\mathcal{R}^n)$ is bounded from below

For every step $k \geq 0$, let $\mathbf{x}^{(k+1)}$ be the next point of the minimising sequence

- Point $\mathbf{x}^{(k+1)}$ is determined from current point $\mathbf{x}^k$ and a vector $\mathbf{d}^{(k)}$
- Vector $\mathbf{d}^{(k)}$ itself depends on two term
  - ⇝ The gradient vector $\nabla f\big[\mathbf{x}^{(k)}\big]$ of function f
  - ⇝ A step-length parameter $\alpha_k \in \mathcal{R}$

Line-search methods (cont.)

The algorithmic formulation of the general line-search/descent method

Pseudo-code

Let $\mathbf{x}^{(0)} \in \mathcal{R}^n$ be an initial minimiser

$$\begin{aligned} \textit{Find direction} \quad & \mathbf{d}^{(k)} \in \mathcal{R}^n \\ \textit{Compute step-length} \quad & \alpha_k \in \mathcal{R} \\ \textit{Set} \quad & \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \end{aligned}$$

Line-search methods (cont.)

Definition

Vector $\mathbf{d}^{(k)}$ must be a ***descent direction***

A descent direction satisfies the following conditions

$$\begin{aligned} \mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big] < 0, & \text{ if } \nabla f\big[\mathbf{x}^{(k)}\big] \neq \mathbf{0} \\ \mathbf{d}^{(k)} = \mathbf{0}, & \text{ if } \nabla f\big[\mathbf{x}^{(k)}\big] = \mathbf{0} \end{aligned} \qquad (12)$$

⇝ $\nabla f\big[\mathbf{x}^{(k)}\big]$ gives the direction of max positive growth of f from $\mathbf{x}^{(k)}$

⇝ $\mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big]$ is the directional derivative of f along $\mathbf{d}^{(k)}$

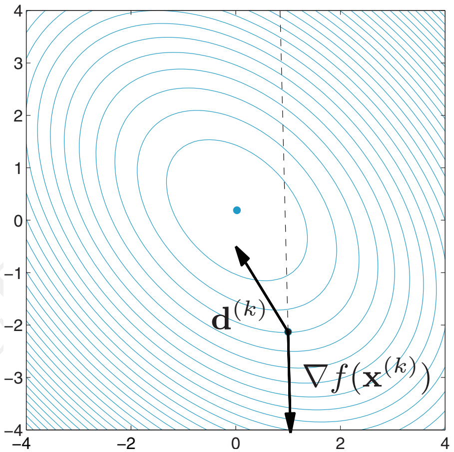
First condition ensures moves in a direction opposite to the gradient

⇝ The iterates move towards a minimiser

Line-search methods (cont.)

Contour lines of function $f(\mathbf{x})$ and its gradient vector evaluated at $\mathbf{x}^{(k)}$

- $\mathbf{d}^{(k)}$ is a suitable descent direction



Optimal value $\alpha_k \in \mathcal{R}$ guarantees max variation of f along $\mathbf{d}^{(k)}$

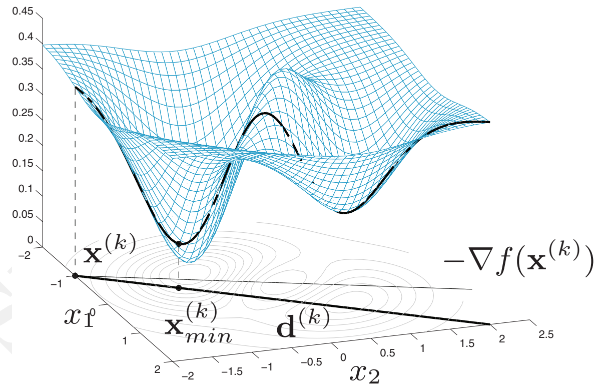
- Once $\mathbf{d}^{(k)}$ is determined

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Line-search methods (cont.)

α_k can be computed by solving a one-dimensional minimisation problem

- Minimise the restriction of $f(\mathbf{x})$ along $\mathbf{d}^{(k)}$
- $\mathbf{x}_{\text{min}}^{(k)}$ is the minimiser along $\mathbf{d}^{(k)}$



The computation of α_k is quite involved (when f is not quadratic)

$\rightsquigarrow$ There are alternative techniques that approximate α_k well

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent directions

Line-search methods

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent directions

Newton's directions

$$\mathbf{d}^{(k)} = -\mathbf{H}^{-1}\big[\mathbf{x}^{(k)}\big]\nabla f\big[\mathbf{x}^{(k)}\big] \qquad (13)$$

- Matrix $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$ is the Hessian matrix at the k-th step
- Vector $\nabla f\big[\mathbf{x}^{(k)}\big]$ is the gradient vector at the k-th step

Quasi-Newton directions

$$\mathbf{d}^{(k)} = -\mathbf{H}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big] \qquad (14)$$

- Matrix $\mathbf{H}_k$ is an approximation of the true Hessian $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$
- It is used when second derivatives are heavy to compute

Descent directions (cont.)

Gradient directions

$$\mathbf{d}^{(k)} = -\nabla f\big[\mathbf{x}^{(k)}\big] \tag{15}$$

- These are quasi-Newton directions, with $\mathbf{H}_k = \mathbf{I}$, $\forall k \geq 0$

Conjugate-gradient directions

$$\begin{aligned} \mathbf{d}^{(0)} &= -\nabla f\big[\mathbf{x}^{(0)}\big] \\ \mathbf{d}^{(k+1)} &= -\nabla f\big[\mathbf{x}^{(k+1)}\big] + \beta_k \mathbf{d}^{(k)}, \quad k \geq 0 \end{aligned} \tag{16}$$

- Coefficients β_k can be chosen according to different criteria

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent directions (cont.)

For all $k \geq 0$, gradient directions are valid descent directions

$$\begin{aligned} \mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big] &< 0, \quad \text{if } \nabla f\big[\mathbf{x}^{(k)}\big] \neq \mathbf{0} \\ \mathbf{d}^{(k)} &= \mathbf{0}, \quad \text{if } \nabla f\big[\mathbf{x}^{(k)}\big] = \mathbf{0}, \end{aligned} \tag{17}$$

Conjugate-gradient directions are valid directions for some suitable β_k

Newton's and quasi-Newton's directions can also be valid directions

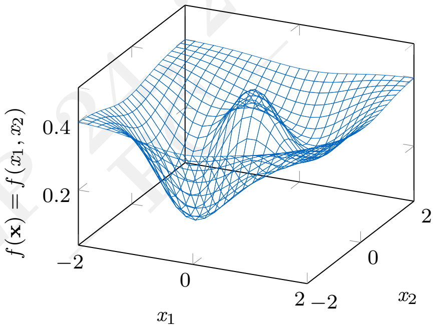
- $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$ and $\mathbf{H}_k$ need be positive definite matrices

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent directions (cont.)

Example

$$f(\mathbf{x}) = 2/5 - 1/10(5x_1^2 + 5x_2^2 + 3x_1x_2 - x_1 - 2x_2)e^{\left[-(x_1^2+x_2^2)\right]}$$



Two local minimisers, one local maximiser and two saddle points

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

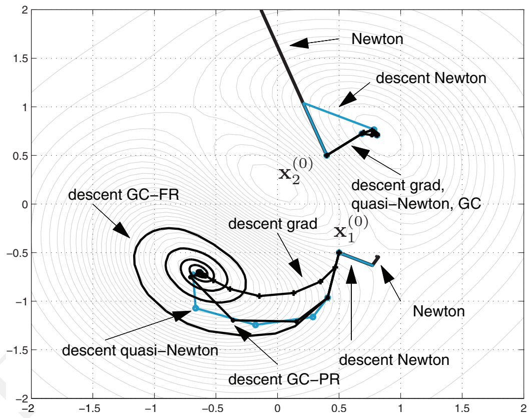
Descent directions (cont.)

We compare sequences $\{\mathbf{x}^{(k)}\}$ from Newton's and descent methods

- Various descent directions
- From $\mathbf{x}_1^{(0)}$ and $\mathbf{x}_2^{(0)}$

Descent directions (cont.)

$\mathbf{x}_1^{(0)} = (0.5, -0.5)$



- Newton's method converges rapidly towards the saddle point
- Newton's directions take a first step identical to Newton's

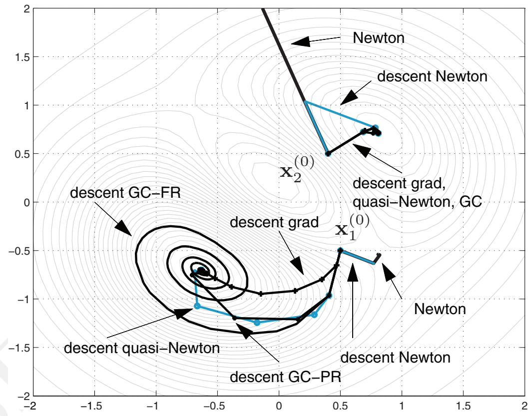
⇝ Then collapse due to a non-positive definite matrix $\mathbf{H}_k$

- Others converge with different speeds into a local minimum
- Fastest convergence by quasi-Newton's directions

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent directions (cont.)

$\mathbf{x}_2^{(0)} = (0.4, 0.5)$



- Newton's method diverges
- Newton's directions converge to a local minimum

⇝ Newton's method and directions share the same first direction

- All others also converge to the same local minimiser

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k

Line-search methods

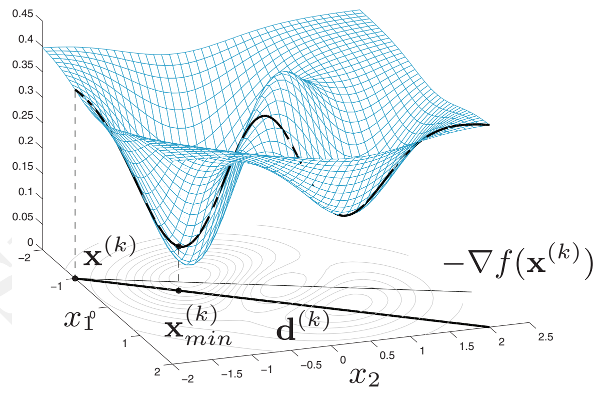
Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k

Let $\mathbf{d}^{(k)}$ be a descent direction

- How to set the step-length α_k

The new iterate $\mathbf{x}^{(k+1)}$ is (should be) the minimiser of f along $\mathbf{d}^{(k)}$



Step-length α_k (cont.)

The new iterate $\mathbf{x}^{(k+1)}$ should be the minimiser of f along $\mathbf{d}^{(k)}$

Choose α_k such that the minimisation is exact

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} f\big[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\big]$$

or (18)

$$f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] = \min_{\alpha \in \mathbb{R}} f\big[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\big]$$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

A second-order Taylor expansion of f around $\mathbf{x}^{(k)}$ yields

$$f\big[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\big] = f\big[\mathbf{x}^{(k)}\big] + \alpha \mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big] + \frac{\alpha^2}{2} \mathbf{d}^{(k)T} \mathbf{H}\big[\mathbf{x}^{(k)}\big] \mathbf{d}^{(k)} + o(||\alpha \mathbf{d}^{(k)}||^2) \quad (19)$$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Remark

Consider the special case in which f is a quadratic function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{x}^T \mathbf{b} + c$$

- $\mathbf{A} \in \mathbb{R}^{n \times n}$ symmetric and positive definite
- $\mathbf{b} \in \mathbb{R}^n$
- $c \in \mathbb{R}$

The expansion is exact, the infinitesimal residual is null

$$f\big[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\big] = f\big[\mathbf{x}^{(k)}\big] + \alpha \mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big] + \frac{\alpha^2}{2} \mathbf{d}^{(k)T} \mathbf{H}\big[\mathbf{x}^{(k)}\big] \mathbf{d}^{(k)} + \cancel{o(||\alpha \mathbf{d}^{(k)}||^2)}$$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

For every $k \geq 0$, we have

$$\begin{aligned} f\big[\mathbf{x}^{(k)}\big] &= \frac{1}{2} \mathbf{x}^{(k)T} \mathbf{A} \mathbf{x}^{(k)} - \mathbf{x}^{(k)T} \mathbf{b} + c \\ \nabla f\big[\mathbf{x}^{(k)}\big] &= \mathbf{A} \mathbf{x}^{(k)} - \mathbf{b} = -\mathbf{r}^{(k)} \\ \nabla^2 f\big[\mathbf{x}^{(k)}\big] &= \mathbf{H}\big[\mathbf{x}^{(k)}\big] = \mathbf{A} \end{aligned}$$

Differentiate $f\big[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\big] = f\big[\mathbf{x}^{(k)}\big] + \alpha \mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big] + \frac{\alpha^2}{2} \mathbf{d}^{(k)T} \mathbf{H}\big[\mathbf{x}^{(k)}\big] \mathbf{d}^{(k)}$ wrt α and set the derivative equal to zero to get $\min_{\alpha \in \mathbb{R}} f[\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}]$

$$\frac{\mathrm{d}}{\mathrm{d}\alpha_k} f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{x}^{(k)}\big] = -\mathbf{d}^{(k)T} \mathbf{r}^{(k)} + \alpha_k \mathbf{d}^{(k)} \mathbf{A} \mathbf{d}^{(k)} = 0$$

(20)

$$\rightsquigarrow \alpha_k = \frac{\mathbf{d}^{(k)T} \mathbf{r}^{(k)}}{\mathbf{d}^{(k)T} \mathbf{A} \mathbf{d}^{(k)}}$$

Step-length α_k (cont.)

Let $\mathbf{d}^{(k)}$ be gradient directions, $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)}) = \mathbf{r}^{(k)}$

⇝ The gradient method for solving linear systems

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Let $\mathbf{d}^{k}$ be conjugate-gradient directions, $\mathbf{d}^{(k+1)} = -\nabla f\left[\mathbf{x}^{(k+1)}\right] + \beta_k \mathbf{d}^{(k)}$

Set,

$$\beta_k = \frac{\left[\mathbf{A}\mathbf{d}^{(k)}\right]^T \mathbf{r}^{(k+1)}}{\mathbf{d}^{(k)T}\mathbf{A}\mathbf{d}^{(k)}} \tag{21}$$

⇝ The conjugate-gradient method for solving linear systems

■

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Let f be a non-quadratic function

The computation of the optimal α_k requires an iterative method

⇝ Numerical solution of minimisation along $\mathbf{d}^{(k)}$

Remark

⇝ Demanding and often not worth it

⇝ Stick with an approximation of α_k

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

How to pick a good approximated value of α_k?

Impose a condition to the new iterate $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

$$\rightsquigarrow f\left[\mathbf{x}^{(k+1)}\right] < f\left[\mathbf{x}^{(k)}\right] \tag{22}$$

Step-length α_k (cont.)

Example

A natural strategy for setting α_k

- Initially assign a large α_k
- Then, reduce it iteratively
- Until, $f\big[\mathbf{x}^{(k+1)}\big] < f\big[\mathbf{x}^{(k)}\big]$ is satisfied

The strategy does not guarantee a $\{\mathbf{x}^k\}$ that converges to $\mathbf{x}^*$

- Steps can be too long (go beyond the minimum)
- Steps can be too short (get infinitesiamal)

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

There exist alternative (better/reliable) criteria for $\alpha_k > 0$

$\rightsquigarrow$ **Wolfe's conditions**

Definition

Let α_k be the step-length

α_k is accepted if

$$\begin{aligned} f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] &\leq f\big[\mathbf{x}^{(k)}\big] + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big] \\ {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] &\geq \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big] \end{aligned} \qquad (23)$$

The two additional parameters, constants σ and δ

- $0 < \sigma < \delta < 1$

$\mathbf{d}^{(k)} \nabla f\big[\mathbf{x}^{(k)}\big]$ *is the directional derivative of f along direction $\mathbf{d}^{(k)}$*

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

$$\begin{aligned} f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] &\leq f\big[\mathbf{x}^{(k)}\big] + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big] \\ {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] &\geq \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big] \end{aligned}$$

First condition (**Armijo's rule**) inhibits too small variations of f

- With respect to step-length and directional derivative

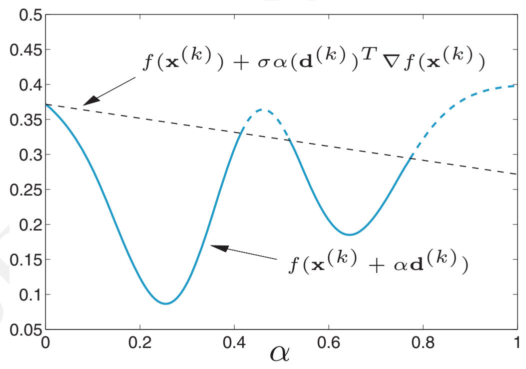
Changes in f need be proportional to step-length and directional derivative

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

The terms in the first of the two Wolfe's conditions, for $\sigma = 0.2$

$$f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \leq f\big[\mathbf{x}^{(k)}\big] + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$



Condition is satisfied for α corresponding to the continuous line

Step-length α_k (cont.)

$$f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \leq f\big[\mathbf{x}^{(k)}\big] + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$
$${\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \geq \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$

Second condition states that the directional derivative of f at new point $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ should be δ times larger than it was at point $\mathbf{x}^{(k)}$

- Point $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$ is a valid candidate if f at such point decreases less than it does at $\mathbf{x}^{(k)}$ (closer to a minimiser)

This second condition prevents steps whose length would be too small

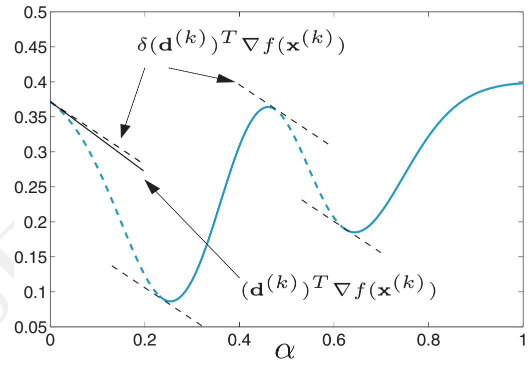
- Happens where f has a largely negative directional derivative

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Lines with slope $\delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$ in second condition, $\delta = 0.9$

$${\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \geq \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$



Condition is satisfied for α corresponding to the continuous line

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Wolfe's conditions are jointly satisfied in the interval

$$0.23 \leq \alpha \leq 0.41 \text{ or } 0.62 \leq \alpha \leq 0.77$$

Values of $\alpha \in [0.62, 0.77]$ are far from the minimiser of f along $\mathbf{d}^{(k)}$

- Also α where the directional derivative is large are accepted

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Definition

Wolfe's strong conditions

$$f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \leq f\big[\mathbf{x}^{(k)}\big] + \sigma \alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$
$$\big|{\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big]\big| \leq -\delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big] \qquad (24)$$

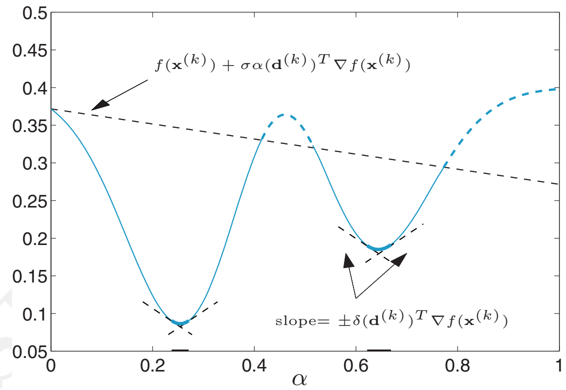
This conditions are more restrictive (duh!)

- The first condition is unchanged
- The second one inhibits f from large variations about $\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

Step-length α_k (cont.)

Acceptable α must belong to small intervals around the minimisers

- (thick continuous arcs)



- For $\sigma = 0.2$ and $\delta = 0.9$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

Remark

Suppose that $f \in \mathbb{C}^2(\mathbb{R}^n)$ is bounded from below in $\{\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}, \alpha > 0\}$

- Let $\mathbf{d}^{(k)}$ be a descent direction at $\mathbf{x}^{(k)}$

It can be shown that for all σ and δ such that $0 < \sigma < \delta < 1$ there exist non-empty intervals of α_k that satisfy Wolfe's weak and strong conditions

In practice[2], σ is usually chosen to be very small (e.g., $\sigma = 10^4$)

Typical values for δ

- $\delta = 0.9$ for Newton, quasi-Newton and gradient directions
- $\delta = 0.1$ for conjugate-gradient directions

[2] J. Nocedal and S. Wrigth (2006): *Numerical optimization.*

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

A strategy for step-lengths α_k that satisfy Wolfe's conditions

Backtracking

- Start with $\alpha = 1$
- Then reduce it by a given factor ρ (tipically, $\rho \in [0.1, 0.5)$)
- Until, the first weak condition is satisfied

For $\mathbf{x}^{(k)}$ and a direction $\mathbf{d}^{(k)}$, for $\sigma \in (0, 1)$ and $\rho \in [0.1, 0.5)$

Pseudo-code

Set $\alpha = 1$
    while $f\big[\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}\big] > f\big[\mathbf{x}^{(k)}\big] + \sigma\alpha\mathbf{d}^{(k)}\nabla f\big[\mathbf{x}^{(k)}\big]$
        $\alpha = \rho\alpha$
    end
Set $\alpha_k = \alpha$

Second condition is never checked, as step-lengths are not small

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Step-length α_k (cont.)

```
function [x,alpha_k]=bTrack(fun,x_k,g_k,d_k,varargin)
%BTRACK Backtracking with line search
% [X,ALPHA_K]=BTRACK(FUN,X_K,G_K,D_K) x_{k+1}=x_k+alpha_k*d_k
%  in the descent method, alpha_k by backtracking with
%  sigma=1e-4 and rho=0.25
%
% [X,ALPHA_K]=BTRACK(FUN,X_K,G_K,D_K,SIGMA,RHO) sigma and rho
%  can be inputed - sigma in (1e-4,0.1) and rho in (0.1,0.5)
%
% FUN is the function handle of the objective function
% X_K is element x_k, G_K is the gradient, D_K is d_k

if nargin == 4
 sigma = 1.0e-4; rho = 1/4;
else
 sigma = varargin {1}; rho = varargin {2};
end

minAlpha = 1.0e-5; % Smallest steplength
alpha_k = 1.0; f_k = fun(x_k);

k = 0; x = x_k + alpha_k*d_k;
while fun(x) > f_k+sigma*alpha_k*g_k'*d_k & alpha_k > minAlpha
 alpha_k = alpha_k*rho;
 x = x_k + alpha_k*d_k; k = k+1;
end
```

Step-length α_k (cont.)

The descent method with various descent directions

- α_k is determined by backtracking

```
%DSCENT Descent method of minimisation
%[X,ERR,ITER]=DSCENT(FUN,GRAD_FUN,X_0,TOL,KMAX,TYP,HESS_FUN)
% Approximates the minimiser of FUN using descent directions
%  Newton (TYP=1), BFGS (TYP=2), GRADIENT (TYP=3), and the
%  CONJUGATE-GRADIENT method with
%   beta_k by Fletcher and Reeves (TYP=41)
%   beta_k by Polak and Ribiere   (TYP=42)
%   beta_k by Hestenes and Stiefel(TYP=43)
%
% Step length is calculated using backtracking (bTrack.m)
%
% FUN, GRAD_FUN and HESS_FUN (TYP=1 only) are function handles
%  for the objective, gradient and Hessian matrix
%  With TYP=2, HESS_FUN approximates the exact Hessian at X_0
%
% TOL is the stop check tolerance
% KMAX is the maximum number of iteration
```

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

```
function [x,err,iter]=dScent(fun,grad_fun,x_0,tol,kmax,typ,varargin)
if nargin>6; if typ==1; hess=varargin{1};
   elseif typ==2; H=varargin{1}; end; end

err=tol+1; k=0; xk=x0(:); gk=grad(xk); dk=-gk; eps2=sqrt(eps);

while err>tol & k<kmax
 if typ==1; H = hess_fun(xk); dk = -H\gk;              % Newton
 elseif typ==2; dk = -H\gk;                              % BFGS
 elseif typ==3; dk = -gk;                            % Gradient
 end
 [xk1,alphak]=bTrack(fun,xk,gk,dk);
 gk1=grad_fun(xk1);
  if typ==2                                       % BFGS update
   yk = gk1-gk; sk = xk1-xk; yks = yk'*sk;
   if yks > eps2*norm(sk)*norm(yk)
    Hs=H*sk; H=H+(yk*yk')/yks-(Hs*Hs')/(sk'*Hs);
   end
  elseif typ>=40                                   % CG upgrade
   if typ==41; betak=(gk1'*gk1)/(gk'*gk);                 % FR
   elseif typ==42; betak=(gk1'*(gk1-gk))/(gk'*gk);        % PR
   elseif typ==43; betak=(gk1'*(gk1-gk))/(dk'*(gk1-gk)); % HS
   end
   dk = -gk1 + betak*dk;
  end
  xk = xk1; gk = gk1; k = 1 + k; xkt = xk1;
 for i=1:length(xk1); xkt(i) = max([abs(xk1(i)),1]); end
  err = norm((gk1.*xkt)/max([abs(fun(xk1)),1]),Inf);
end
x = xk; iter = k;
if (k==kmax & err>tol); disp('[KMAX]'); end
```

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with Newton's directions

Line-search methods

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with Newton's directions

Let us consider a descent method with Newton's directions

⇝ Newton directions

$$\mathbf{d}^{(k)} = -\mathbf{H}^{-1}\big[\mathbf{x}^{(k)}\big]\nabla f\big[\mathbf{x}^{(k)}\big]$$

Let step-lengths α_k satisfy Wolfe's conditions

⇝ Wolfe step lengths α_k

$$f\big(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big) \le f\big[\mathbf{x}^{(k)}\big] + \sigma\alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$
$${\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \ge \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$

Let $f \in \mathbb{C}^2(\mathbb{R}^n)$ bounded from below

Descent method with Newton's directions (cont.)

Pseudo-code

Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$

Compute step $\alpha_k \in \mathbb{R}$

Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with Newton's directions (cont.)

Suppose that the Hessian $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$ is symmetric, for all $k \geq 0$

$\rightsquigarrow$ (from the assumption on f)

Suppose that $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$ is also positive definite (no uphill moves)

Let $\mathbf{B}_k = \mathbf{H}\big[\mathbf{x}^{(k)}\big]$

Suppose that $\exists M > 0 : K(\mathbf{B}_k) = ||\mathbf{B}_k||_2 ||\mathbf{B}_k^{-1}||_2 \leq M$, for all $k \geq 0$

- $K(\mathbf{B}_k)$ is the (one) **spectral condition number** of $\mathbf{B}_k$
- Uniform upper bound on the condition number

Then, Newton's sequence $\big\{\mathbf{x}^{(k)}\big\}$ converges to a stationary point $\mathbf{x}^*$

$\rightsquigarrow$ By letting $\alpha_k = 1$ for $k \geq \overline{k}$, the converge is quadratic

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with Newton's directions (cont.)

Definition

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be a matrix*

Consider the problem of finding a scalar λ *(complex or real) and a non-null vector* $\mathbf{x} \in \mathbb{C}^n$ *such that*

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

Any λ *that satisfy this equation is an* ***eigenvalue*** *of* $\mathbf{A}$

- $\mathbf{x}$ *is the corresponding* ***eigenvector***

The ***spectral condition number*** *of* $\mathbf{A}$ *is the quantity*

$$K(\mathbf{A}) = \frac{\lambda_{max}}{\lambda_{min}}$$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with Newton's directions (cont.)

Remark

If Hessians are positive definite, $\mathbf{x}^*$ cannot be a maximiser or saddle point

- The stationary point must necessarily be a minimiser

It can happen that $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$ is not positive definite for some point $\mathbf{x}^{(k)}$

- $\mathbf{d}^{(k)}$ may not be a descent direction
- Wolfe's conditions might become meaningless

Descent method with Newton's directions (cont.)

Hessian can transformed to make them positive definite

$$\mathbf{B}_k = \mathbf{H}\big[\mathbf{x}^{(k)}\big] + \mathbf{E}_k$$

- $\mathbf{E}_k$ is some suitable matrix (either diagonal or full)
- $\mathbf{E}_k$ is such that $\mathbf{d}^{(k)} = -\mathbf{B}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big]$ is a descent direction

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton's directions

Line-search methods

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton

Let us consider a descent method with quasi-Newton directions

- Quasi-Newton directions

$$\mathbf{d}^{(k)} = -\mathbf{H}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big]$$

⇝ $\mathbf{H}_k$ approximates the true Hessian $\mathbf{H}\big[\mathbf{x}^{(k)}\big]$

Let step-lengths α_k satisfy Wolfe's conditions

⇝ Wolfe step lengths α_k

$$f\big(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big) \leq f\big[\mathbf{x}^{(k)}\big] + \sigma\alpha_k {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$
$${\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}\big] \geq \delta {\mathbf{d}^{(k)}}^T \nabla f\big[\mathbf{x}^{(k)}\big]$$

Let $f \in \mathbb{C}^2(\mathbb{R}^n)$ bounded from below

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton's directions (cont.)

Suppose we are given a symmetric and positive definite matrix $\mathbf{H}_0$

⇝ How do we build matrices $\mathbf{H}_k$?

There exists a popular technique used for solving nonlinear systems

⇝ The recursive **Broyden's rank-one update**

Descent method with quasi-Newton's directions (cont.)

Matrices $\mathbf{H}_k$ are required the satisfy certain conditions

- They must satisfy the secant condition

$$\mathbf{H}_{k+1}\big[\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\big] = \nabla f\big[\mathbf{x}^{(k+1)}\big] - \nabla f\big[\mathbf{x}^{k}\big]$$

- They must be symmetric, as $\mathbf{H}(\mathbf{x})$
- They must be positive definite, $\mathbf{d}^{(k)}$ are descent
- They must satisfy

$$\lim_{k\to\infty} \frac{\big|\big|\big[\mathbf{H}_k - \mathbf{H}(\mathbf{x}^*)\big]\mathbf{d}^{(k)}\big|\big|}{\big|\big|\mathbf{d}^{(k)}\big|\big|} = 0$$

This ensures that $\mathbf{H}_k$ is a good approximation of $\mathbf{H}\big[\mathbf{x}^*\big]$ along the descent direction $\mathbf{d}^{(k)}$ and guarantees a super-linear rate of convergence

Descent method with quasi-Newton's directions (cont.)

Definition

*A strategy by Broyden, Fletcher, Goldfarb and Shanno (**BFGS**)*

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\mathbf{x}^{(k)^T}\mathbf{s}^{(k)}} - \frac{\mathbf{H}_k\mathbf{s}^{(k)}\mathbf{s}^{(k)^T}\mathbf{H}_k^T}{\mathbf{s}^{(k)^T}\mathbf{H}_k\mathbf{s}^{(k)}} \qquad (25)$$

- $\mathbf{s}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$
- $\mathbf{y}^{k} = \nabla f\big[\mathbf{x}^{(k+1)}\big] - \nabla f\big[\mathbf{x}^{(k)}\big]$

Descent method with quasi-Newton's directions (cont.)

Matrices $\mathbf{H}_{k+1}$ are symmetric and positive definite under condition

$$\mathbf{y}^{(k)^T}\mathbf{s}^{(s)} > 0$$

It is satisfied when step lengths α_k are either weak or strong Wolfe

Descent method with quasi-Newton's directions (cont.)

BFGS is a descent method, with quasi-Newton $\mathbf{d}^{(k)}$ and Wolfe's α_ks

$\rightsquigarrow$ $\mathbf{d}^{(k)} = -\mathbf{H}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big]$

$\rightsquigarrow$

$$f\big(\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}\big) \leq f\big[\mathbf{x}^{(k)}\big] + \sigma\alpha_k\mathbf{d}^{(k)^T}\nabla f\big[\mathbf{x}^{(k)}\big]$$
$$\mathbf{d}^{(k)^T}\nabla f\big[\mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}\big] \geq \delta\mathbf{d}^{(k)^T}\nabla f\big[\mathbf{x}^{(k)}\big]$$

Pseudo-code

Let $\mathbf{x}^{(0)}$ *be an initial solution*

Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$

Compute step length $\alpha_k \in \mathbb{R}$

Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$

Descent method with quasi-Newton's directions (cont.)

Pseudo-code

Let $\mathbf{x}^{(0)}$ *be an initial solution*

Let $\mathbf{H}_0 \in \mathbb{R}^{n \times n}$ *be a suitable symmetric and positive definite matrix*

$\rightsquigarrow$ $\mathbf{H}_0 \in \mathbb{R}^{n \times n}$ *approximates* $\mathbf{H}[\mathbf{x}^{(0)}]$

Solve $\mathbf{H}_k \mathbf{d}^{(k)} = -\nabla \mathbf{f}[\mathbf{x}^{(k)}]$

Compute α_k *that satisfies Wolfe's conditions*

Set

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \\ \mathbf{s}^{(k)} &= \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} &= \nabla f[\mathbf{x}^{(k+1)}] - \nabla f[\mathbf{x}^{(k)}] \end{aligned}$$

Compute $\mathbf{H}_{k+1} = \mathbf{H}_k + \dfrac{\mathbf{y}^{(k)} {\mathbf{y}^{(k)}}^T}{{\mathbf{x}^{(k)}}^T \mathbf{s}^{(k)}} - \dfrac{\mathbf{H}_k \mathbf{s}^{(k)} {\mathbf{s}^{(k)}}^T \mathbf{H}_k^T}{{\mathbf{s}^{(k)}}^T \mathbf{H}_k \mathbf{s}^{(k)}}$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton's directions (cont.)

Remark

The cost of calculating $\mathbf{d}^{(k)}$ is $\mathcal{O}(n^3)$, at every iteration $k \geq 0$

- Can be reduced to $\mathcal{O}(n^2)$ by using recursive QR on $\mathbf{H}_k$

Setting $\mathbf{H}_0 = \mathbf{I}$ gives faster convergence to $\mathbf{x}^*$

$\rightsquigarrow$ Some experimental evidence, only

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton's directions (cont.)

Example

Rosenbrock's function

$$f(\mathbf{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

Let $\varepsilon = 10^{-6}$ be the tolerance

```
x_0 = [+1.2; -1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;

options = optimset ('LargeScale','off');    % Switches to BFGS
[xstar,fval,exitflag,output] = fminunc(fun,x_0,options)
```

Convergence after 24 iterations and 93 function evaluations

We did not input an expression for evaluating the gradient

- It was, silently, approximated
- (finite difference methods)

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with quasi-Newton's directions (cont.)

We can define and input the analytical gradient

```
x_0 = [+1.2; -1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;
grad_fun = @(x)[-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1)); ...
                 +200*(x(2)-x(1)^2)];

options = optimset('LargeScale','off','GradObj','on');
[xstar,fval,exitflag,output] = fminunc({fun,grad_fun},...
                                        x_0,options)
```

Convergence after 25 iterations and 32 function evaluations

Descent method with quasi-Newton's directions (cont.)

Remark

In Octave, BFGS is implemented by the M-command `bfgsmin`

- M-command `fminunc` implements a different method
- (A trust-region method)

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Descent method with gradient and conjugate-gradient directions

Line-search methods

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Gradient and conjugate-gradient directions

Let us first consider the general descent method

Pseudo-code

Find direction $\mathbf{d}^{(k)} \in \mathbb{R}^n$

Compute step $\alpha_k \in \mathbb{R}$

Set $\mathbf{x}^{(\mathbf{k+1})} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$

The gradient (descent) directions

$$\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$$

If $f \in \mathbb{C}^2(\mathbb{R}^n)$ is bounded from below and step lengths α_k are Wolfe

$\leadsto$ This method converges (linearly) to a stationary point

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Gradient and conjugate-gradient directions (cont.)

Let us now consider conjugate directions

$$\mathbf{d}^{(0)} = -\nabla f(\mathbf{x}^{(0)})$$
$$\mathbf{d}^{(k+1)} = -\nabla f(\mathbf{x}^{(k+1)}) - \beta_k \mathbf{d}^{(k)}, \quad k \geq 0$$

There are several options for setting β_k

Gradient or conjugate-gradient directions (cont.)

⇝ **Fletcher-Reeves**

$$\beta_k^{FR} = -\frac{||\nabla f[\mathbf{x}^{(k)}]||^2}{||\nabla f[\mathbf{x}^{(k-1)}]||^2} \quad (26)$$

⇝ **Polak-Ribière (-Polyak)**

$$\beta_k^{PR} = -\frac{\nabla f[\mathbf{x}^{(k)}]^T \{\nabla f[\mathbf{x}^{(k)}] - \nabla f[\mathbf{x}^{(k-1)}]\}}{||\nabla f[\mathbf{x}^{(k-1)}]||^2} \quad (27)$$

⇝ **Hestenes-Stiefel**

$$\beta_k^{HS} = -\frac{\nabla f[\mathbf{x}^{(k)}]^T \{\nabla f[\mathbf{x}^{(k)}]^T - \nabla f[\mathbf{x}^{(k-1)}]\}}{\mathbf{d}^{(k-1)^T} \{\nabla f[\mathbf{x}^{(k)}] - \nabla f[\mathbf{x}^{(k-1)}]\}} \quad (28)$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Gradient and conjugate-gradient directions (cont.)

Remark

Suppose true the condition that f is quadratic and strictly convex

Then, all the aforementioned options are equivalent

$$\rightsquigarrow \beta_k = \frac{[\mathbf{Ad}^{(k)}]^T \mathbf{r}^{(k+1)}}{\mathbf{d}^{(k)^T} \mathbf{Ad}^{(k)}}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Trust-region methods

Numerical optimisation

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Trust-region methods

Line search methods are designed to set first the descent direction $\mathbf{d}^{(k)}$

- Then, they determine the step-length α_k

These steps are performed at each k-th step

Trust-region methods simultaneously choose direction and step length

This is done by building a ball of radius δ_k centred at $\mathbf{x}^{(k)}$

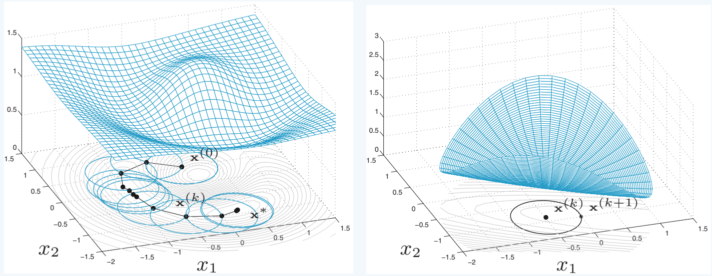
- The ball is the **trust region**, at iteration k

Within the ball, a quadratic approximation $\tilde{f}_k$ of f is computed

- The new $\mathbf{x}^{(k+1)}$ is the minimiser of $\tilde{f}_k$ in the trust region

Trust-region methods (cont.)

Example



Convergence history and quadratic approximation $\tilde{f}_k$ at step $k = 8$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

To compute $\tilde{f}_k$, we start with some trust radius $\delta_k > 0$

- Determine a second-order Taylor expansion of f about $\mathbf{x}^{(k)}$

$$\tilde{f}_k(\mathbf{s}) = f\left[\mathbf{x}^{(k)}\right] + \mathbf{s}\nabla f\left[\mathbf{x}^{(k)}\right] + \frac{1}{2}\mathbf{s}^T\mathbf{H}_k\mathbf{s}, \quad \forall \mathbf{s} \in \mathbb{R}^n \tag{29}$$

  $\mathbf{H}_k$ is either the Hessian of f at $\mathbf{x}^{(k)}$ or a suitable approximation

- We then compute the solution $\mathbf{s}^{(k)}$

$$\mathbf{s}^{(k)} = \underset{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||\leq\delta_k}{\arg\min} \quad \tilde{f}_k(\mathbf{s}) \tag{30}$$

⇝ At this stage, we also compute the quantity

$$\rho_k = \frac{f\left[\mathbf{x}^{(k)} + \mathbf{s}^{(k)}\right] - f\left[\mathbf{x}^{(k)}\right]}{\tilde{f}_k\left[\mathbf{s}^{(k)}\right] - \tilde{f}_k(\mathbf{0})} \tag{31}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

$$\rho_k = \frac{f\left[\mathbf{x}^{(k)} + \mathbf{s}^{(k)}\right] - f\left[\mathbf{x}^{(k)}\right]}{\tilde{f}_k\left[\mathbf{s}^{(k)}\right] - \tilde{f}_k(\mathbf{0})}$$

A comparison between variation of f and variation of $\tilde{f}_k$

⇝ From point $\mathbf{x}^{(k)}$ to point $\mathbf{x}^{(k)} + \mathbf{s}^{(k)}$

If ρ_k is about one, the approximation is considered to e good

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

If ρ_k is approximately one, we accept $\mathbf{s}^{(k)}$ and move on to next iteration

⇝ We set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$

- (however, if the minimiser of $\tilde{f}_k$ lie on the boundary of the trust region, we extend the latter before proceeding to next iteration)

If ρ_k is either negative or positive (and much smaller than one)

⇝ We reduce the ball's size and calculate a new $\mathbf{s}^{(k)}$

$$\mathbf{s}^{(k)} = \underset{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||\leq\delta_k}{\arg\min} \quad \tilde{f}_k(\mathbf{s})$$

If ρ_k is much larger than one, we accept $\mathbf{s}^{(k)}$ and keep the trust region

⇝ Then we move to the next iteration

Trust-region methods (cont.)

Remark

Consider the situation in which second derivatives of f are available

We could set $\mathbf{H}_k$ to be equal to the Hessian

- (or a variant, if not positive definite)

Otherwise, $\mathbf{H}_k$ can be built recursively

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

Let $\mathbf{H}_k$ be symmetric positive definite and let $\big|\big|\mathbf{H}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big]\big|\big| \leq \delta_k$

- Then, $\mathbf{s}^{(k)} = \mathbf{H}_k^{-1}\nabla f\big[\mathbf{x}^{(k)}\big]$ is a minimiser
- It is within the trust region

Otherwise, the minimiser of $\tilde{f}_k$ lies outside the trust region

$\rightsquigarrow$ We must solve the minimisation of $\tilde{f}_k$

- Constrained to the δ_k-ball at $\mathbf{x}^{(k)}$

$$\min_{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||=\delta_k} \tilde{f}_k(\mathbf{s}) \tag{32}$$

This is a constrained optimisation problem

$\rightsquigarrow$ We can use the Lagrange multipliers

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

At each iteration k, we look for the minimiser of the Lagrangian function

$$\mathcal{L}(\mathbf{s},\lambda) = \tilde{f}_k(\mathbf{s}) + 1/2\lambda(\mathbf{s}^T\mathbf{s} - \delta_k)$$

To be optimised with respect to both $\mathbf{s}$ and the regularisation term λ

We search for a vector $\mathbf{s}^{(k)}$ and a scalar $\lambda^{(k)} > 0$ satisfying the system

$$\begin{aligned} \big[\mathbf{H}_k + \lambda^{(k)}\mathbf{I}\big]\mathbf{s}^{(k)} &= -\nabla f\big[\mathbf{x}^{(k)}\big] \\ \big[\mathbf{H}_k + \lambda^{(k)}\mathbf{I}\big] &\text{ is PSD} \\ \big|\big|\mathbf{s}^{(k)}\big|\big| - \delta_k &= 0 \end{aligned} \tag{33}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

From $\big[\mathbf{H}_k + \lambda^{(k)}\mathbf{I}\big]\mathbf{s}^{(k)} = -\nabla f\big[\mathbf{x}^{(k)}\big]$, we compute $\mathbf{s}^{(k)} = \mathbf{s}^{(k)}\big[\lambda^{(k)}\big]$

We substitute it in $||\mathbf{s}^{(k)}|| - \delta_k = 0$

$$\rightsquigarrow \varphi\big[\lambda^{(k)}\big] = \frac{1}{\big|\big|\mathbf{s}^{(k)}\big[\lambda^{(k)}\big]\big|\big|} - \frac{1}{\delta_k} = 0$$

The non-linear equation in λ is equivalent to system (33)

- It can be solved using Newton's method

Trust-region methods (cont.)

For some given λ_0, set $\mathbf{g}^{(k)} = \nabla f\big[\mathbf{x}^{(k)}\big]$

Pseudo-code

For $l = 0, 1, \dots$ (typically, less than 5 iterations are needed)

Compute $\mathbf{s}_l^{(k)} = -\big[\mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}\big]^{-1}\mathbf{g}^{(k)}$

Evaluate $\varphi\big[\lambda_l^{(k)}\big] = 1/||\mathbf{s}_l^{(k)}|| - 1/\delta_k$

Evaluate $\varphi'\big[\lambda_l^{(k)}\big]$

Compute $\lambda_{l+1}^{(k)} = \lambda_l^{(k)} - \varphi\lambda_l^{(k)}/\varphi'\big[\lambda_l^{(k)}\big]$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Trust-region methods (cont.)

Vector $\mathbf{s}_l^{(k)}$ is obtained by **Cholesky factorisation** of $\big[\mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}\big]$

- Provided that matrix $\mathbf{B}^{(k)} = \mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}$ is positive definite
- If $\mathbf{B}^{(k)}$ is symmetric (definition of $\mathbf{H}_k$), its eigenvalues are all real

Remark

Usually, a regularised matrix $\mathbf{B}_l^{(k)} + \beta\mathbf{I}$ is used instead of $\mathbf{B}^{(k)}$

- β should be larger than the negative eigenvalue of $\mathbf{B}^{(k)}$ of largest modulus

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Trust-region methods (cont.)

Definition

Cholesky factorisation

Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric and positive definite matrix

$$\mathbf{A} = \mathbf{R}^T\mathbf{R}$$

$\mathbf{R}$ *is upper triangular with positive elements on the diagonal*

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2

Trust region methods (cont.)

For $\mathbf{g}^{(k)} = \nabla f\big[\mathbf{x}^{(k)}\big]$ and for some given δ_k

Pseudo-code

Solve $\mathbf{H}_k\mathbf{s} = -\mathbf{g}^{(k)}$ *(means* $\mathbf{s} = -\mathbf{H}_k^{-1}\mathbf{g}^{(k)}$*)*

If $||\mathbf{s}|| \leq \delta_k$ *and* $\mathbf{H}_k$ *is positive definite*
  Set $\mathbf{s}^{(k)} = \mathbf{s}$
else
  Let β_1 *be the negative eigenvalue of* $\mathbf{H}_k$ *with largest modulus*
  Set $\lambda_0^{(k)} = 2|\beta_1|$
  For $l = 0, 1, \dots$
    Compute $\mathbf{R} : \mathbf{R}^T\mathbf{R} = \mathbf{H}_k + \lambda_l^{(k)}\mathbf{I}$
    Solve $\mathbf{R}^T\mathbf{R}\mathbf{s} = \mathbf{g}^{(k)}$, $\mathbf{R}^T\mathbf{q} = \mathbf{s}$
    Update $\lambda_{l+1}^{(k)} = \lambda_l^{(k)} + \big(||\mathbf{s}||/||\mathbf{q}||\big)^2 \dfrac{||\mathbf{s}|| - \delta_k}{\delta_k}$
  Set $\mathbf{s}^{(k)} = \mathbf{s}$
endif

Trust-region methods (cont.)

For a fast convergence, a good radius δ_k is truly fundamental

The criterion for accepting a solution $\mathbf{s}^{(k)}$ is based on a comparison

- The variation of f and that of its quadratic approximation $\tilde{f}_k$

Remark

As $\mathbf{x}^{(k)}$ moves to $\mathbf{x}^{(k)} + \mathbf{s}^{(k)}$

$$\rho_k = \frac{f\left[\mathbf{x}^{(k)} + \mathbf{s}^{(k)}\right] - f\left[\mathbf{x}^{(k)}\right]}{\tilde{f}_k\left[\mathbf{s}^{(k)}\right] - \tilde{f}_k(\mathbf{0})}$$

If $\rho_k \approx 1$

- $\mathbf{s}^{(k)}$ is accepted, the ball is enlarged if the minimum is on the boundary

If $\rho_k \approx 0$ or $\rho_k < 0$

- $\mathbf{s}^{(k)}$ is not accepted and the ball is diminished

Trust-region methods (cont.)

Let $\mathbf{x}^{(0)}$ be an initial solution

Let the initial radius of the ball be $\delta_0 \in (0, \hat{\delta})$ with maximum radius $\hat{\delta} > 0$

Let $\{\eta_1, \eta_2, \gamma_1, \gamma_2\}$ be the four real parameters for updating the ball

- $0 < \eta_1 < \eta_2 < 1$
- $0 < \gamma_1 < 1 < \gamma_2$

Let $0 \leq \mu \leq \eta_1$ be the real parameter for accepting a solution

...

Trust-region methods (cont.)

Then, for $k = 0, 1, \dots$ until convergence

Pseudo-code

Compute $f\left[\mathbf{x}^{(k)}\right]$, $\nabla f\left[\mathbf{x}^{(k)}\right]$ *and* $\mathbf{H}_k$

Solve $\min_{\mathbf{s} \in \mathbf{R}^n : ||\mathbf{s}||_2 \leq \delta_k} \tilde{f}_k(\mathbf{s})$

Compute ρ_k

If $\rho_k > \mu$
    Set $\mathbf{x}^{(x+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$
else
    Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$
endif

If $\rho_k < \eta_1$
    Set $\delta_{k+1} = \gamma_1 \delta_k$
elseif $\eta_1 \leq \rho_k \leq \eta_2$
    Set $\delta_{k+1} = \delta_k$
elseif $\rho_k > \eta_2$ *and* $||\mathbf{s}^{(k)}|| = \delta_k$
    Set $\delta_{k+1} = \min\{\gamma_2 \delta_k, \hat{\delta}\}$
endif

Trust-region methods (cont.)

Choice of parameters[3]

- ⇝ $\eta_1 = 1/4$
- ⇝ $\eta_2 = 3/4$
- ⇝ $\gamma_1 = 1/4$
- ⇝ $\gamma_2 = 8/4$

- By choosing $\mu = 0$, we accept any step yielding a decrease of f
- By choosing $\mu > 0$, we accept steps for which the variation of f is at least μ times the variation of its quadratic model $\tilde{f}_k$

[3] J. Nocedal and S. Wrigth (2006): *Numerical optimization.*

Trust-region methods (cont.)

```matlab
%TREGION Trust region optimisation method
%[X,ERR,ITER]=TREGION(FUN,GRAD_FUN,X_0,DELTA_0, ...
% TOL,KMAX,TYP,HESS_FUN)
% Approximates the minimiser of FUN with gradient GRAD_FUN
%
% If TYP=1 Hessian is inputed as HESS_FUN
% If TYP NE 1 Hessian is rank-one approximated
%
% FUN and GRAD_FUN (and HESS_FUN) are function handles
% X_0 is the initial  point
% TOL is stop check tolerance
% DELTA_0 is initial radius of trust ball
% KMAX are maximum number of iterations
```

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

```matlab
function [x,err,iter]= tRegion(fun,grad_fun,x_0,delta_0, ...
                               tol,kmax,typ,hess_fun)

delta = delta_0; err = 1 + tol; k = 0; mu = 0.1; delta_m = 5;
eta_1 = 0.25; eta_2 = 0.75; gamma_1 = 0.25; gamma_2 = 2.00;

xk = x_0(:); gk = grad_fun(xk); eps2 = sqrt(eps);
if typ==1; Hk=hess_fun(xk); else; Hk=eye(length(xk)); end

while err > tol & k < kmax
 [s]=trust_one(Hk,gk,delta);
 rho=(fun(xk+s)-fun(xk))/(s'*gk+1/2*s'*Hk*s);
 if rho > mu; xk1 = xk + s; else; xk1 = xk; end
 if rho < eta_1; delta = gamma_1*delta;
 elseif rho > eta_2 & abs(norm(s)-delta) < sqrt(eps)
  delta=min([gamma_2*delta,delta_m]);
 end
 gk1 = grad_fun(xk1);
 err = norm((gk1.*xk1)/max([abs(fun(xk1)),1]),Inf);
 if typ == 1; xk = xk1; gk = gk1; Hk = hess_fun(xk);  % Newton
 else                                                  % quasi-Newton
  gk1 = grad(xk1); yk = gk1-gk; sk=xk1-xk; yks = yk'*sk;
  if yks > eps_2*norm(sk)*norm(yk)
   Hs = Hk*sk; Hk = Hk+(yk*yk')/yks-(Hs*Hs')/(sk'*Hs);
  end
  xk = xk1; gk = gk1;
 end
 k=k+1;
end

x = xk; iter = k;
if (k==kmax & err>tol); disp('Accuracy not met [KMAX]'); end
```

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

```matlab
function [s] = trust_one (Hk,gk,delta)
maxiter=5;

s = -Hk\gk; d = eigs(Hk,1,'sa'); % 1st smallest algebraic evalue

if norm(s) > delta | d<0
 lambda = abs(2*d); I = eye(size(Hk));
 for l=1:maxiter
  R = chol(lambda*I+Hk);
  s = -R\(R'\gk); q = R'\s;
  lambda = lambda+(s'*s)/(q'*q)*(norm(s)-delta)/delta;
  if lambda < -d
   lambda = abs(2*lambda);
  end
 end
end
```

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

Example

Approximate the minimiser of function

$$f(x_1, x_2) = 7/5 + \frac{x_1 + 2x_2 + 2x_1x_2 - 5x_1^2 - 5x_2^2}{[5\exp(x_1^2 + x_2^2)]}$$

Use the trust-region method

A local maximum, a saddle point and two local minima

- The local minima are near $(-1.0, +0.2)$ and $(+0.3, -0.9)$
- The second minimum is the global one

Trust region methods (cont.)

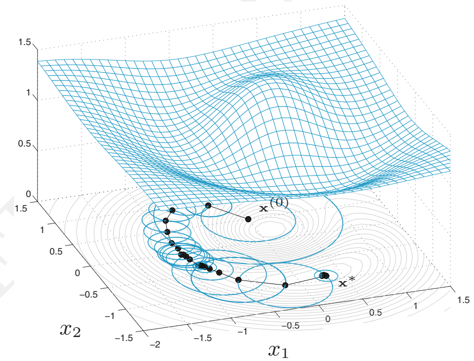
```
fun = @(x) (x(1)+2*x(2)+2*x(1)*x(2)-5*x(1)^2-5*x(2)^2) / ...
           (5*exp(x(1)^2+x(2)^2)) + 7.5;

grad_fun = @(x) [(1 + 2*x(2)-10*x(1)-2*x(1)*(x(1)+2*x(2) + ...
                      2*x(1)*x(2)-5*x(1)^2-5*x(2)^2)) / ...
                      (5*exp(x(1)^2+x  (2)^2));
                 (2 + 2*x(1)-10*x(2)-2*x(2)*(x(1)+2*x(2) + ...
                      2*x(1)*x(2)-5*x(1)^2-5*x(2)^2)) / ...
                      (5*exp(x(1)^2+x(2)^2))];

delta_0 = 0.5; x_0 = [0.0;0.5];
tol = 1e-5; kmax = 100; imax=5;
typ = 2;

[x,er,it]=tRegion(fun,grad_fun,x_0,delta_0,tol,kmax,typ,imax)
```

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

Trust-region, approximated Hesse matrix

⇝ 24 iterations, $\mathbf{x}^* \approx (+0.28, -0.90)$

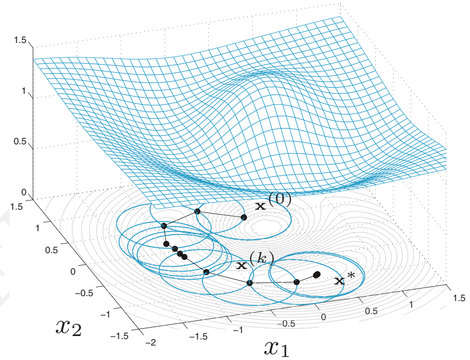


Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust-region methods (cont.)

Trust-region, exact Hessian

⇝ 12 iterations



■

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Trust region methods (cont.)

Example

Rosenbrock's function

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

```
fun = @(x) (1-x(1))^2+100*(x(2)-x(1)^2)^2;
grad_fun = @(x)[-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1)); ...
                200*(x(2)-x(1)^2)];

x_0=[+1.2;-1.0];

options = optimset ('LargeScale','on'); % Trust-region
options = optimset ('GradObj','on');    % Gradient

[x,fval,exitflag,output]=fminunc({fun,grad_fun},x_0,options)
```

Trust-region (Matlab)

⇝ 8 iterations, 9 function evaluations

■

Trust-region methods (cont.)

Remark

The M-command `fminunc` in Octave implements the trust region method

- With approximated Hessians $\mathbf{H}_k$, computed with BFGS

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)^T}}{\mathbf{x}^{(k)^T}\mathbf{s}^{(k)}} - \frac{\mathbf{H}_k\mathbf{s}^{(k)}\mathbf{s}^{(k)^T}\mathbf{H}_k^T}{\mathbf{s}^{(k)^T}\mathbf{H}_k\mathbf{s}^{(k)}}$$

The option `'LargeScale'` is not used

Non-linear least-squares

Numerical optimisation

Non-linear least-squares

The **least-squares method** is often used for approximating either functions $f(x)$ or sets of data $\{(x_k, y_k), k = 0, \dots, K\}$ by some function $\tilde{f}$

- Often $\tilde{f}$ depends linearly on a set of coefficients $\{a_j, j = 1, \dots, m\}$

Example

$$\tilde{f}(x|\{a_j\}_{j=0}^m) = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m$$

The coefficients $\{a_j\}_{j=0}^m$ are unknown

They must be determined from data

$$\{(x_k, y_k), k = 0, \dots, K\}$$

$$\rightsquigarrow \min_{\{a_j, j=1,\dots,m\}} \sum_{k=0}^{K} \Big[y_k - \underbrace{\tilde{f}(x_k|\{a_j\})}_{a_0 + a_1 x_k + a_2 x_k^2 + \cdots + a_m x_k^m} \Big]^2$$

This problem is called a **least-squares** problem

The problem becomes nonlinear when $\tilde{f}$ non-linearly depends on $\{a_j\}$

Non-linear least-squares (cont.)

Definition

Let $\mathbf{R}(\mathbf{x}) = \big[r_1(\mathbf{x}), \dots, r_n(\mathbf{x})\big]^T$ *with* $r_i : \mathbb{R}^m \to \mathbb{R}$ *be some smooth function*

We want to find

$$\min_{\mathbf{x} \in \mathbb{R}^m} f(\mathbf{x}), \quad \text{with } f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x}) = \frac{1}{2}\big|\big|\mathbf{R}(\mathbf{x})\big|\big|^2 \tag{34}$$

We assume that $n \geq m$

If functions $r_i(\mathbf{x})$ are non-linear, then function $f(\mathbf{x})$ may not be convex

$\rightsquigarrow$ Thus, $f(\mathbf{x})$ may have multiple stationary points

We can use Newton, descent directions and trust-region methods

Non-linear least-squares (cont.)

Consider the special form of f

We have assembled the components $r_i(\mathbf{x})$ into a residual vector

$$\mathbf{R}(\mathbf{x}) = \big[r_1(\mathbf{x}), \dots, r_n(\mathbf{x})\big]^T$$

Because of this, we compactly rewrote the objective function

$$f(\mathbf{x}) = \frac{1}{2}\big|\big|\mathbf{R}(\mathbf{x})\big|\big|^2$$

Non-linear least-squares (cont.)

The derivatives of $f(\mathbf{x})$ can be expressed in terms of the Jacobian of $\mathbf{R}$

⇝ Partial derivatives of $r_i(\mathbf{x})$ with respect to x_j

$$\mathbf{J}_{\mathbf{R}}(\mathbf{x}) = \Big[\frac{\partial r_i}{\partial x_j}\Big]_{\substack{j=1,\dots,m \\ i=1,\dots,m}} = \begin{bmatrix} \Big[\frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_m}\Big] \\ \Big[\frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_2}{\partial x_m}\Big] \\ \vdots & \vdots & \ddots & \vdots \\ \Big[\frac{\partial r_n}{\partial x_1} & \frac{\partial r_n}{\partial x_2} & \cdots & \frac{\partial r_n}{\partial x_m}\Big] \end{bmatrix} = \begin{bmatrix} \nabla r_1(\mathbf{x})^T \\ \nabla r_2(\mathbf{x})^T \\ \vdots \\ \nabla r_n(\mathbf{x})^T \end{bmatrix}$$

Non-linear least-squares (cont.)

Gradient and Hessian of the cost function can be compactly written

$$\begin{aligned} \nabla f(\mathbf{x}) &= \sum_{i=1}^{n} r_i(\mathbf{x}) \nabla r_i(\mathbf{x}) = \mathbf{J}_{\mathbf{R}}(\mathbf{x})^T \mathbf{R}(\mathbf{x}) \\ \nabla^2 f(\mathbf{x}) &= \mathbf{J}_{\mathbf{R}}(\mathbf{x})^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}) + \sum_{i=1}^{n} r_i(\mathbf{x}) \nabla r_i(\mathbf{x}) = \mathbf{J}_{\mathbf{R}}(\mathbf{x})^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}) + \mathbf{S}(\mathbf{x}) \end{aligned} \qquad (35)$$

⇝ The second derivatives of $\mathbf{R}$ cannot be calculated from the Jacobian

$$\mathbf{S}_{lj}(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial^2 r_i}{\partial x_l \partial x_j}(\mathbf{x}) r_i(\mathbf{x}), \text{ for } l, j = 1, \dots, m$$

Non-linear least-squares (cont.)

Calculation of the Hesse matrix can be heavy when m and n are large

- This is mostly due to matrix $\mathbf{S}(\mathbf{x})$

In some cases, $\mathbf{S}(\mathbf{x})$ is less influent than $\mathbf{J}_{\mathbf{R}}(\mathbf{x})^T \mathbf{J}_{\mathbf{R}}(\mathbf{x})$

⇝ It could be approximated or neglected

⇝ It simplifies the construction of $\mathbf{H}(\mathbf{x})$

We discuss two methods devoted to handling such cases

Gauss-Newton method

Nonlinear least-squares

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method

The **Gauss-Newton method** is a variant of the Newton method

Given $\mathbf{x}^{(0)} \in \mathbb{R}^n$, for $k = 0, 1, \dots$ until convergence

Pseudo-code

Solve $\mathbf{H}\big[\mathbf{x}^{(k)}\big]\boldsymbol{\delta}\mathbf{x}^{(k)} = -\boldsymbol{\nabla} f\big[\mathbf{x}^{(k)}\big]$

Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$

The Hessian $\mathbf{H}(\mathbf{x})$ is approximated by neglecting $\mathbf{S}(\mathbf{x})$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Given $\mathbf{x}^{(0)} \in \mathbb{R}^m$ and for $k = 0, 1, \dots$ until the convergence

Pseudo-code

Solve $\big\{\mathbf{J_R}(\mathbf{x}^k)^T\mathbf{J_R}\big[\mathbf{x}^{(k)}\big]\big\}\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{J_R}\big[\mathbf{x}^{(k)}\big]^T\mathbf{R}\big[\mathbf{x}^{(k)}\big]$

Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$

The system in the first equation may have infinitely many solutions

If $\mathbf{J_R}\big[\mathbf{x}^{(k)}\big]$ is not full rank

- ⇝ Stagnation
- ⇝ Non-convergence
- ⇝ Convergence to a non-stationary point

If $\mathbf{J_R}\big[\mathbf{x}^{(k)}\big]$ is full rank, the linear system has form $\mathbf{A}^T\mathbf{A}\mathbf{x}^* = \mathbf{A}^T\mathbf{b}$

- It can be solved by using QR or SVD factorisations of $\mathbf{J_R}(\mathbf{x})$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

```
function [x,err,iter]=nllsGauNewtn(r,jr,x_0,tol,kmax,varargin)
%NLLSGAUNEW Nonlinear least-squares with Gauss-Newton method
% [X,ERR,ITER]=NLLSGAUNEW(R,JR,X_0,TOL,KMAX)
% R and JR: Function handles for objective R and its Jacobian
% X_0 is the initial solution
% TOL is the stop check tolerance
% KMAX is the max number of iterations

err = 1 + tol; k = 0;
xk = x_0(:);

rk = r(xk,varargin{:}); jrk = jr(xk,varargin{:});

while err > tol & k < kmax
 [Q,R] = qr(jrk,0); dk = -R\(Q'*rk);
 xk1 = xk + dk;
 rk1 = r(xk1,varargin{:});
 jrk1 = jr(xk1,varargin{:});

 k = 1 + k; err = norm(xk1 - xk);
 xk = xk1; rk = rk1; jrk = jrk1;
end

x = xk; iter = k;

if (k==kmax & err > tol)
 disp('nllsGauNewtn stopped w\o reaching accuracy [KMAX]');
end
```

The Gauss-Newton method (cont.)

Remark

Neglecting $\mathbf{S}(\mathbf{x}^{(k)})$ at step k amounts to approximating $\mathbf{R}(\mathbf{x})$

The first-order Taylor expansion of $\mathbf{R}(\mathbf{x})$ at $\mathbf{x}^*$

$$\tilde{\mathbf{R}}_k(\mathbf{x}) = \mathbf{R}\big[\mathbf{x}^{(k)}\big] + \mathbf{J}_{\mathbf{R}}\big[\mathbf{x}^{(k)}\big]\big[\mathbf{x} - \mathbf{x}^{(k)}\big] \tag{36}$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Convergence of the method is not always guaranteed

- It depends on f and initial solution

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Let $\mathbf{x}^*$ be a stationary point for $f(\mathbf{x})$

Let $\mathbf{J}_{\mathbf{R}}(\mathbf{x})$ be full rank in a suitable neighbourhood of $\mathbf{x}^*$

Then,

If $\mathbf{S}(\mathbf{x}^*) = 0$ (if $\mathbf{R}(\mathbf{x})$ is linear or $\mathbf{R}(\mathbf{x}^*) = \mathbf{0}$)

1. The Gauss-Newton method is locally quadratically convergent
2. It coincides with the Newton's method

If $||\mathbf{S}(\mathbf{x}^*)||_2$ is small compared to the smallest positive eigenvalue of

$$\mathbf{J}_{\mathbf{R}}(\mathbf{x}^*)^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}^*)$$

1. (e.g., when $\mathbf{R}(\mathbf{x})$ is mildly non-linear or its residual $\mathbf{R}(\mathbf{x}^*)$ is small)
2. Gauss-Newton converges linearly

If $||\mathbf{S}(\mathbf{x})||_2$ is large compared to the smallest positive eigenvalue of

$$\mathbf{J}_{\mathbf{R}}(\mathbf{x}^*)^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}^*)$$

1. Gauss-Newton may not converge, even if $\mathbf{x}^{(0)}$ is very close to $\mathbf{x}^*$
2. (e.g., when $\mathbf{R}(\mathbf{x})$ is strongly non-linear or its residual $\mathbf{R}(\mathbf{x}^*)$ is large)

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Remark

Line-search can be used in combination with Gauss-Newton

- Replace $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)}$ with $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \boldsymbol{\delta}\mathbf{x}^{(k)}$
- Computation of step-lengths α_k is as per usual

If $\mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})$ is full rank, matrix $\mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})^T \mathbf{J}_{\mathbf{R}}(\mathbf{x}^{(k)})$ is symmetric and PD

- $\boldsymbol{\delta}\mathbf{x}^{(k)}$ is a descent direction for $f(\mathbf{x})$

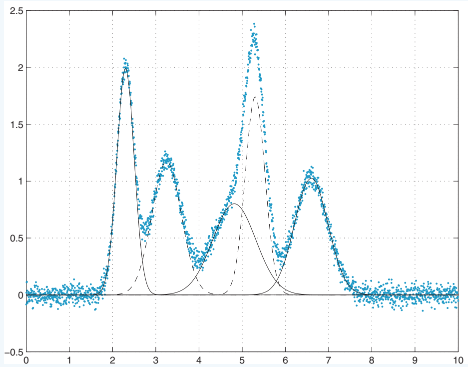
Under suitable assumptions on $f(\mathbf{x})$, we get the globally convergent method

$\rightsquigarrow$ **Damped Gauss-Newton method**

The Gauss-Newton method (cont.)

Example

Compress an audio signal to a set of parameters

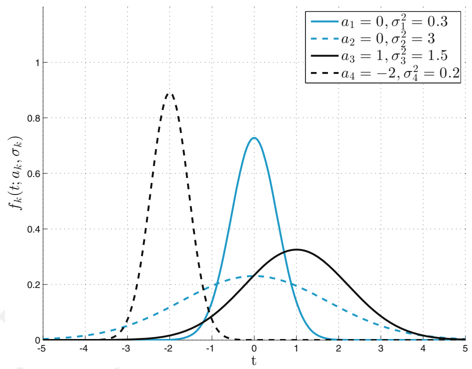


The signal intensity is modelled as a sum of m Gaussian functions

$$f_k(t|a_k,\sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\exp\Big[-\frac{(t-a_k)^2}{2\sigma_k^2}\Big],\quad t\in[t_0,t_F], k=1,\dots,m$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)



Each peak or component is characterised by two coefficients

- The centre, a_k
- The (square of the) spread, σ_k^2

$$f(t|\mathbf{a},\boldsymbol{\sigma}) = \sum_{k=1}^{m} f_k(t; a_k,\sigma_k)$$

- $\mathbf{a} = [a_1,\cdots,a_k]$
- $\boldsymbol{\sigma} = [\sigma_1,\cdots,\sigma_k]$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Find $\mathbf{a}$ and $\boldsymbol{\sigma}$ that minimise the residual sum of squares

$$\min_{\mathbf{a},\boldsymbol{\sigma}} \sum_{i=1}^{n}\Big[f(t_i|\mathbf{a},\boldsymbol{\sigma}) - y_i\Big]^2$$

From recorded audio intensities y_i at sampling times t_i

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Generate $n = 2000$ time-intensity pairs $(t_i, y_i)_{i=1}^{n}$ with $t_i \in (0, 10)$

$\rightsquigarrow$ The sum of 5 Gaussian components

$$f_k(t|a_k,\sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\exp\Big[-\frac{(t-a_k)^2}{2\sigma_k^2}\Big]$$

- Plus some little random noise

```
a = [2.3, 3.2, 4.8, 5.3, 6.6]; m = length(a);
sigma = [0.2, 0.3, 0.5, 0.2, 0.4];

gComp = @(t,a,sigma) exp(-((t-a)/(sigma*sqrt(2))).^2)/ ...
                     (sigma*sqrt(pi*2));

n = 2000; t = linspace(0,10,n)'; y = zeros(n,1);
for k=1:m
 y = y + gComp(t,a(k),sigma(k));
end

y = y + 0.05*randn(n,1);               % Little additive noise
```

The Gauss-Newton method (cont.)

Solve the nonlinear least-squares problem of form

$$\min_{\mathbf{x}\in\mathbb{R}^m} \Phi(\mathbf{x}), \quad \text{with } \Phi(\mathbf{x}) = \frac{1}{2}\big|\big|\mathbf{R}(\mathbf{x})\big|\big|^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x})$$

$r_i(\mathbf{x}) = f(t_i|\mathbf{a},\boldsymbol{\sigma}) - y_i = \sum_{k=1}^{m} f_k(t_i|a_k,\sigma_k) - y_i$

We also have,

$$\frac{\partial r_i}{\partial a_k} = f_k(t_i|a_k,\sigma_k)\Big[\frac{t_i - a_k}{\sigma_k}\Big]$$
$$\frac{\partial r_i}{\partial \sigma_k} = f_k(t_i|a_k,\sigma_k)\Big[\frac{(t_i - a_k)^2}{\sigma_k^3} - \frac{1}{2\sigma_k}\Big]$$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

Gauss-Newton

M-command `nllsGauNewtn` (22 iterations)

```
x_0 = [2.0,3.0,4.0,5.0,6.0,0.3,0.3,0.6,0.3,0.3];

tol = 3.0e-5;
kmax = 200;

[x,err,iter]=nllsGauNew(@gmR,@gmJR,x_0,tol,kmax,t,y)

x_a = x(1:m);
x_sigma = x(m+1:end);

h = 1./(x_sigma*sqrt(2*pi));
w = 2*x_sigma*sqrt(log(4));
```


Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

The Gauss-Newton method (cont.)

```
function [R]=gmR(x,t,y)

x = x(:); m = round(0.5*length(x));
a = x(1:m); sigma = x(m+1: end );

gauFun = @(t,a,sigma) [exp(-((t-a)/(sigma*sqrt(2))).^2) ...
                        /(sigma*sqrt(pi*2))];

n = length(t); R = zeros(n,1);
for k = 1:m; R = R + gauFun(t,a(k),sigma(k)); end
R = R - y;
```

```
function [Jr]=gmJR(x,t,y)
x = x(:); m = round(0.5*length(x));
a = x(1:m); sigma = x(m+1: end);

gauFun = @(t,a,sigma) [exp(-((t-a)/(sigma*sqrt(2))).^2) ...
                        /(sigma*sqrt(pi*2))];

n = length(t); JR = zeros(n,2*m); fk = zeros(n,m);
for k = 1:m; fk(:,k) = gauFun(t,a(k),sigma(k)); end
for k = 1:m; JR(:,k) = (fk(:,k).*(t-a(k))/sigma(k)^2)'; end
for k = 1:m
 JR(:,k+m) = (fk(:,k).*((t-a(k)).^2/(k)^3-1/(2*sigma(k))))';
end
```

■

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Levenberg-Marquardt

Nonlinear least-squares

Levenberg-Marquardt

Levenberg-Marquardt is a trust-region method

$$\min_{\mathbf{x}\in\mathbb{R}^m} f(\mathbf{x}), \quad \text{with } f(\mathbf{x}) = \frac{1}{2}||\mathbf{R}(\mathbf{x})||^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(\mathbf{x})$$

We can use the general trust-region formulation

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Levenberg-Marquardt (cont.)

Pseudo-code

Compute $f\left[\mathbf{x}^{(k)}\right]$, $\nabla f\left[\mathbf{x}^{(k)}\right]$ *and* $\mathbf{H}_k$

Solve $\min_{||\mathbf{s}||_2\leq\delta_k} \tilde{f}_k(\mathbf{s})$

Compute ρ_k
If $\rho_k > \mu$
  Set $\mathbf{x}^{(x+1)} = \mathbf{x}^{(k)} + \mathbf{s}^{(k)}$
else
  Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$
endif

If $\rho_k < \eta_1$
  Set $\delta_{k+1} = \gamma_1\delta_k$
elseif $\eta_1 \leq \rho_k \leq \eta_2$
  Set $\delta_{k+1} = \delta_k$
elseif $\rho_k > \eta_2$ *and* $||\mathbf{s}^{(k)}|| = \delta_k$
  Set $\delta_{k+1} = \min\{\gamma_2\delta_k, \hat{\delta}\}$
endif

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Levenberg-Marquardt (cont.)

At each step k, we solve

$$\min_{\mathbf{s}\in\mathbb{R}^n:||\mathbf{s}||\leq\delta_k} \tilde{f}_k(\mathbf{s}), \quad \text{with } \tilde{f}_k(\mathbf{s}) = \frac{1}{2}||\mathbf{R}[\mathbf{x}^{(k)}] + \mathbf{J_R}[\mathbf{x}^{(k)}]\mathbf{s}||^2 \qquad (37)$$

$\tilde{f}_k(\mathbf{x})$ is a quadratic approximation of $f(\mathbf{x})$ about $\mathbf{x}^{(k)}$

⇝ By approximating $\mathbf{R}(\mathbf{x})$ with its linear model

$$\tilde{\mathbf{R}}_k(\mathbf{x}) = \mathbf{R}[\mathbf{x}^{(k)}] + \mathbf{J_R}[\mathbf{x}^{(k)}][\mathbf{x} - \mathbf{x}^{(k)}]$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Levenberg-Marquardt (cont.)

Often $\mathbf{J_R}(\mathbf{x})$ is not full rank, yet the method is well-posed

The method is suited for minimisation problems with strong non-linearities or large residuals $f(\mathbf{x}^*) = 1/2||\mathbf{R}(\mathbf{x}^*))||^2$ about the local minimiser $\mathbf{x}^*$

Remark

Hessian approximations are those of the Gauss-Newton method

The two methods share the same local convergence properties

Convergence rates when Levenberg-Marquardt iterations do converge

- Convergence rate is quadratic, if residual is small at local minimiser
- Convergence rate is linear, otherwise

Derivative-free methods

Unconstrained optimisation

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Derivative-free methods

We describe two simple numerical methods

- **Minimisation of univariate real-valued functions**
- **Minimisation of multivariate real-valued functions**
- (along a single direction)

We then describe the **Nelder and Mead method**

- **Minimisation of functions of several variables**

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation

Derivative-free methods

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation

Let $f : (a, b) \rightarrow \mathbb{R}$ be a continuous function with unique minimiser

$$x^* \in (a, b)$$

Set $I_0 = (a, b)$, for $k \geq 0$ generate a sequence of intervals I_k

$$I_k = (a^{(k)}, b^{(k)})$$

The intervals I_k are of decreasing length and each contains x^*

Golden section and quadratic interpolation (cont.)

For any given k, the next interval I_{k+1} can be determined

1) Let $c^{(k)}, d^{(k)} \in I_k$ with $c^{(k)} < d^{(k)}$ be two points such that

$$\frac{b^{(k)} - a^{(k)}}{d^{(k)} - a^{(k)}} = \frac{d^{(k)} - a^{(k)}}{b^{(k)} - d^{(k)}} = \varphi \qquad (38a)$$

$$\frac{b^{(k)} - a^{(k)}}{b^{(k)} - c^{(k)}} = \frac{b^{(k)} - c^{(k)}}{c^{(k)} - a^{(k)}} = \varphi \qquad (38b)$$

Let φ be the **golden ratio** $\varphi = \dfrac{1+\sqrt{5}}{2} \simeq 1.628$

Golden section and quadratic interpolation (cont.)

2) Using Equation 38a and 38b, we find point $c^{(k)}$ and point $d^{(k)}$

$$c^{(k)} = a^{(k)} + \frac{1}{\varphi^2}(b^{(k)} - a^{(k)}) \qquad (39a)$$

$$d^{(k)} = a^{(k)} + \frac{1}{\varphi}(b^{(k)} - a^{(k)}) \qquad (39b)$$

They are symmetrically placed about the mid-point of I_k

$$\frac{a^{(k)} + b^{(k)}}{2} - c^{(k)} = d^{(k)} - \frac{a^{(k)} + b^{(k)}}{2} \qquad (40)$$

Remark

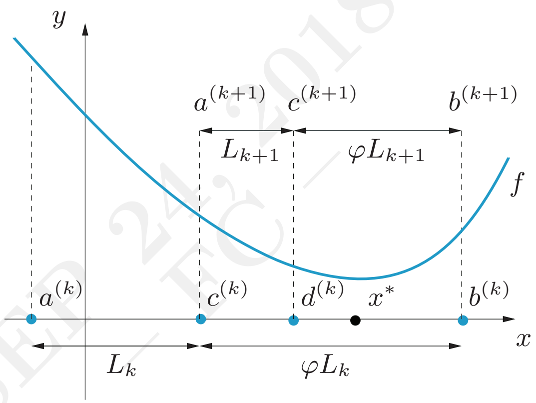
Replace $c^{(k)}$ and $d^{(k)}$ in Equation (40)

Divide by the common factor $\dfrac{b^{(k)} - a^{(k)}}{\varphi^2}$

Get the identity

$$\varphi^2 - \varphi - 1 = 0$$

Golden section and quadratic interpolation (cont.)



The generic iteration of the **golden-section method**

- φ is the golden ratio, while $L_k = c^{(k)} - a^{(k)}$

Golden section and quadratic interpolation (cont.)

Set $a^{(0)} = a$ and $b^{(0)} = b$, the golden section method formulates as

Pseudo-code

For $k = 0, 1, \dots$ until convergence

Compute $c^{(k)}$ and $d^{(k)}$ through Equation (39)

If $f(c^{(k)}) \geq f(d^{(k)})$
set $I_{k+1} = (a^{(k+1)}, b^{(k+1)}) = (c^{(k)}, b^{(k)})$
else
set $I_{k+1} = (a^{(k+1)}, b^{(k+1)}) = (a^{(k)}, d^{(k)})$
endif

It follows that

- If $I_{k+1} = (c^{(k)}, b^{(k)})$, then $c^{(k+1)} = d^{(k)}$
- If $I_{k+1} = (a^{(k)}, d^{(k)})$, then $d^{(k+1)} = c^{(k)}$

Golden section and quadratic interpolation (cont.)

We need to set a **stopping criterion**

When the normalised size of the k-th interval is smaller than a tolerance ε

$$\frac{b^{(k+1)} - a^{(k+1)}}{|c^{(k+1)}| + |d^{(k+1)}|} < \varepsilon \tag{41}$$

The mid-point of the last interval I_{k+1} can be taken as solution

- This is an **approximation of the minimiser** $\mathbf{x}^*$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

By using Equation (38a) and (38b), yields the expression

$$|b^{(k+1)} - a^{(k+1)}| = \frac{1}{\varphi}|b^{(k)} - a^{(k)}| = \cdots = \frac{1}{\varphi^{k+1}}|b^{(0)} - a^{(0)}| \tag{42}$$

The golden-section method converges linearly with rate

$$\varphi^{-1} \simeq 0.618$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

```
function [xmin,fmin,iter]=gSection(fun,a,b,tol,kmax,varargin)
%GSECTION finds the minimum of a function
% XMIN=GSECTION(FUN,A,B,TOL,KMAX) approximates a min point of
%  function FUN in [A,B] by using the golden section method
% If the search fails, an error message is returned
% FUN can be i) an inline function, ii) an anonymous function
%  or iii) a function defined in a M-file
% XMIN=GSECTION(FUN,A,B,TOL,KMAX,P1,P2,...) passes parameters
%  P1, P2,... to function FUN(X,P1,P2,...)
% [XMIN,FMIN,ITER]= GSECTION(FUN,...) returns the value of FUN
%  at XMIN and number of iterations ITER done to find XMIN

phi = (1+sqrt(5))/2;
iphi(1) = inv(phi); iphi(2) = inv(1+phi);
c = iphi(2)*(b-a) + a; d = iphi(1)*(b-a) + a;
err = 1+tol; k = 0;

while err > tol & k < kmax
 if(fun(c) >= fun(d))
  a = c; c = d; d = iphi(1)*(b-a) + a;
 else
  b = d; d = c; c = iphi(2)*(b-a) + a;
 end
 k = 1 + k; err = abs(b-a)/(abs(c)+abs(d));
end

xmin = 0.5*(a+b); fmin = fun(xmin); iter = k;
if (iter == kmax & err > tol)
 fprintf('The method stopped after reaching the maximum number
          of iterations, and without meeting the tolerance');
end
```

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

- `fun` is either an anonymous or an inline function for function f
- `a` and `b` are endpoints of the search interval
- `tol` is the tolerance ε
- `kmax` is the maximum allowed number of iterations

- `xmin` contains the value of the minimiser
- `fmin` is the minimum value of f in (a, b)
- `iter` is the number of iterations carried out by the algorithm

Golden section and quadratic interpolation (cont.)

Example

Evolution of an isolated culture of 250 bacteria (Verhulst model)

$$f(t) = \frac{2500}{1 + 9e^{(-t/3)}}, \quad \text{for } t > 0$$

t denotes time (in days)

Find after how many days population growth rate is maximum

⇝ Where (when?) does function $g(t) = -f'(t)$ has its minimum

$$g(t) = -7500 \frac{\exp(t/3)}{\left[\exp(t/3) + 9\right]^2}$$

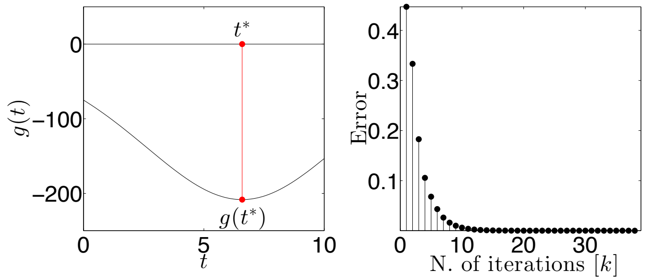
Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Function $g(t)$ admits a global minimiser in $[6, 7]$

```
g = @(t) [-(7500*exp(t/3)) / (exp(t/3)+9)^2];

a = 0; b = 10;
tol = 1.0e-8; kmax = 100;

[tmin gmin,iter]= gSection(g,a,b,tol,kmax);
```

Golden section: 38 iterations, $t^* \approx 6.59$ and $g(t^*) \approx -208$



■

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

The **quadratic interpolation method** is often used as alternative

- Let f be a continuous and convex function
- Let $x^{(0)}$, $x^{(1)}$ and $x^{(2)}$ be three distinct points

We build a sequence of points $x^{(k)}$ with $k \geq 3$ such that

- $x^{(k+1)}$ is the vertex (and thus the minimiser) of the parabola $p_2^{(k)}$
- $p_2^{(k)}$ interpolates f at (node points) $x^{(k)}$, $x^{(k-1)}$ and $x^{(k-2)}$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Definition

*For $k \geq 2$, the order-2 **Lagrange polynomial** at such nodes*

$$\begin{aligned} p_2^{(k)}(x) = & f\left[x^{(k-2)}\right] + \\ & f\left[x^{(k-2)}, x^{(k-1)}\right]\left[x - x^{(k-2)}\right] + \\ & f\left[x^{(k-2)}, x^{(k-1)}, x^{(k)}\right]\left[x - x^{(k-2)}\right]\left[x - x^{(k-1)}\right] \end{aligned}$$

Golden section and quadratic interpolation (cont.)

$$
\begin{aligned}
p_2^{(k)}(x) = & f\big[x^{(k-2)}\big] + \\
& f\big[x^{(k-2)}, x^{(k-1)}\big]\big[x - x^{(k-2)}\big] + \\
& f\big[x^{(k-2)}, x^{(k-1)}, x^{(k)}\big]\big[x - x^{(k-2)}\big]\big[x - x^{(k-1)}\big]
\end{aligned}
$$

In the order-2 Lagrange polynomial $p_2^{(k)}$ for $k \geq 2$, consider the quantities

$$
\begin{aligned}
f\big[x_i, x_j\big] &= \frac{f(x_j) - f(x_i)}{x_j - x_i} \\
f\big[x_i, x_j, x_k\big] &= \frac{f\big[x_j, x_l\big] - f\big[x_i, x_j\big]}{x_l - x_i}
\end{aligned} \tag{43}
$$

The **Newton divided differences**

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Theorem

Consider $n+1$ distinct points

$$
\Big\{\big[x_i, y_i(x_i)\big]\Big\}_{n=0}^{n+1}
$$

There exists only one polynomial $\Pi_n \in \mathbb{P}_n$ of order n or smaller that interpolates them

$$
\Pi_n(x_i) = y_i, \quad \forall i = 0, \dots, n
$$

Π_n is said to be the ***interpolating polynomial*** *of f, if $y_i = f(x_i)$*

$\rightsquigarrow$ *(for some continuous function f)*

It is denoted by $\Pi_n f$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Definition

Consider the components of the Lagrangian basis associated to nodes $\{x_i\}_{i=0}^n$

$$
\varphi_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad i = 0, \dots, n
$$

They are polynomials such that $\{\varphi_i\}$ is the only basis of $\mathbb{P}_n$ satisfying

$$
\varphi_i(x) \in \mathbb{P}_n, \varphi_i(x_j) = \delta_{ij} = \begin{cases} 1, \text{if } i = j \\ 0, \text{otherwise} \end{cases}
$$

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Definition

The ***Lagrange polynomial*** *is the interpolating polynomial $\Pi_n(x)$*

$$
\Pi_n(x) = \sum_{i=0}^{n} y_i \varphi_i(x)
$$

It is expressed in Lagrange form, or wrt the Lagrange basis

$$
\Pi_n(x_i) = \sum_{j=0}^{n} y_j \varphi_j(x_i) = \sum_{j=0}^{n} y_j \delta_{ij} = y_i, \quad i = 0, \dots, n
$$

Golden section and quadratic interpolation (cont.)

By solving the first-order equation ${p'}_2^{(k)}\big[x^{(k+1)}\big] = 0$, we get

$$x^{(k+1)} = \frac{1}{2}\Big\{x^{(k-2)} + x^{(k-1)} - \frac{f\big[x^{(k-2)}, x^{(k-1)}\big]}{f\big[x^{(k-2)}, x^{(k-1)}, x^{(k)}\big]}\Big\} \qquad (44)$$

Next point in the sequence, by setting to zero the derivative of $p_2^{(k)}(x)$

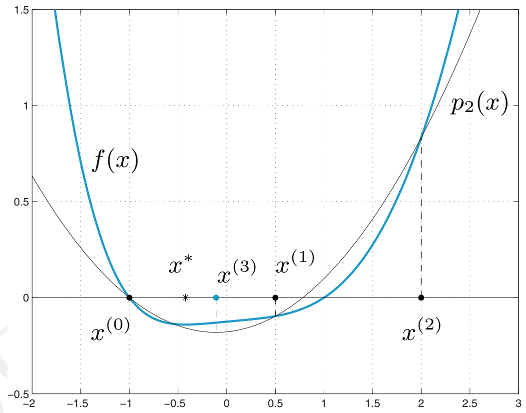
We iterate until $\big|x^{(k+1)} - x^{k}\big| < \varepsilon$, for some tolerance $\varepsilon > 0$

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)



The first step of the quadratic interpolation method

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Example

$$g(t) = -7500 \frac{\exp(t/3)}{\big[\exp(t/3) + 9\big]^2}$$

`fminbnd` combines golden section and parabolic interpolation

```
g = @(t) [-(7500*exp(t/3))/(exp(t/3)+9)^2];

a = 0.0; b = 10.0;
tol = 1.0e-8; kmax = 100;

optionsQ = optimset('TolX', 1.0e-8)
[tminQ,gminQ,exitflagQ,outputQ] = fminbnd(g,a,b,optionsQ);
```

Unconstrained optimisation

UFC/DC
CK0031/CK0248
2018.2

Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Golden section and quadratic interpolation (cont.)

Quadratic interpolation

8 iterations, $t^* \approx 6.59$ and $f(t^*) \approx -208$

- `optimset` sets the tolerance value in structure `optionsQ`
- `qminQ` contains the evaluation of f at the minimiser `tminQ`
- `exitflagQ` indicates the termination state
- `outputQ` has number of iterations and function evaluations

■

Golden section and quadratic interpolation (cont.)

The golden section and the quadratic interpolation method

- They are genuinely one-dimensional techniques
- They can be used to solve multidimensional optimisation problems
- They need be restricted to search along one dimensional directions

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead

Derivative-free methods

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead

Let $n > 1$ and $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous function

Definition

*The **n-simplex** with $n+1$ vertices $\mathbf{x}_i \in \mathbb{R}^n$ for $i = 0, \dots, n$*

$$S = \Big\{\mathbf{y} \in \mathbb{R}^n : \mathbf{y} = \sum_{i=0}^{n} \lambda_i \mathbf{x}_i, \text{ with } \lambda_i \geq 0 : \sum_{i=0}^{n} \lambda_i = 1\Big\} \qquad (45)$$

Intrinsic assumption: Linearly independent vectors $\{(\mathbf{x}_i - \mathbf{x}_0)\}_{i=1}^n$

S is a segment in $\mathbb{R}$, it is a triangle in $\mathbb{R}^2$ and a tetrahedron in $\mathbb{R}^3$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

The **Nelder and Mead method** is a derivative-free minimisation method

- It generates a sequence of simplices $\{S^{(k)}\}_{k \geq 0}$ in $\mathbb{R}^n$

The simplices either run after or circumscribe the minimiser $\mathbf{x}^* \in \mathbb{R}^n$ of f

The method uses simple operations

1. Evaluations of f at the simplices' vertices
2. Geometrical transformations (reflections, expansions, contractions)

Nelder and Mead (cont.)

- At the k-th iteration, the 'worst' vertex of simplex $S^{(k)}$ is identified

$$\mathbf{x}_M^{(k)}, \text{ such that } f\big[\mathbf{x}_M^{(k)}\big] = \max_{0 \leq i \leq n} f\big[\mathbf{x}_i^{(k)}\big]$$

- $\mathbf{x}_M^{(k)}$ is substituted with a new point at which f takes a smaller value
- The new point is got by reflecting/expanding/contracting the simplex along the line joining $\mathbf{x}_M^{(k)}$ and the centroid of the other vertices

$$\mathbf{x}_c^{(k)} = \frac{1}{n} \sum_{\substack{i=0 \\ i \neq M}}^{n} \mathbf{x}_i^{(k)}$$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

SEP 24_FC_2018

How to generate the initial simplex $S^{(0)}$

We take a point $\tilde{\mathbf{x}} \in \mathbb{R}^n$ and a positive real number η

Then, we set

$$\mathbf{x}_i^{(0)} = \tilde{\mathbf{x}} + \eta \mathbf{e}_i, \text{ with } i = 1, \ldots, n$$

$\{\mathbf{e}_i\}$ are the vectors of the standard basis in $\mathbb{R}^n$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

SEP 24_FC_2018

While $k \geq 0$ and until convergence, select the 'worst' vertex of $S^{(k)}$

$$\mathbf{x}_M^{(k)} = \max_{0 \leq i \leq n} f\big[\mathbf{x}_i^{(k)}\big] \tag{46}$$

Then, replace it by a new point to form the new simplex $S^{(k+1)}$

Unconstrained optimisation
UFC/DC CK0031/CK0248 2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

SEP 24_FC_2018

The new point is chosen by firstly selecting

$$\begin{aligned} \mathbf{x}_m^{(k)} &= \min_{0 \leq i \leq n} f\big[\mathbf{x}_i^{(k)}\big] \\ \mathbf{x}_\mu^{(k)} &= \max_a f\big[\mathbf{x}_i^{(k)}\big] \end{aligned} \tag{47}$$

and secondly by defining the **centroid** point

$$\overline{\mathbf{x}}^{(k)} = \frac{1}{n} \sum_{\substack{i=0 \\ i \neq M}}^{n} \mathbf{x}_i^{(k)} \tag{48}$$

This is the centroid of hyperplane $H^{(k)}$ passing through vertices $\{\mathbf{x}_i\}_{\substack{i=0 \\ i \neq M}}^{n}$

Nelder and Mead (cont.)

Thirdly, compute reflection $\mathbf{x}_\alpha^{(k)}$ of $\mathbf{x}_M^{(k)}$ with respect to hyperplane $H^{(k)}$

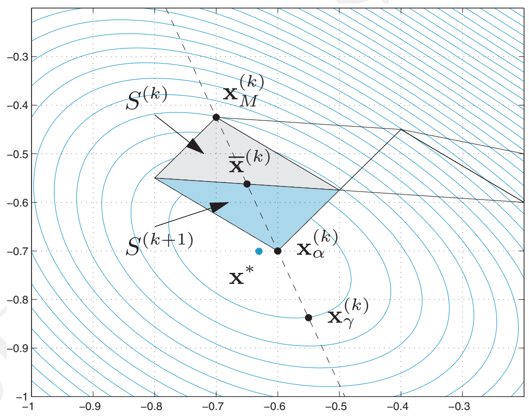
$$\mathbf{x}_\alpha^{(k)} = (1-\alpha)\overline{\mathbf{x}}^{(k)} + \alpha \mathbf{x}_M^{(\alpha)} \tag{49}$$

The **reflection coefficient** $\alpha < 0$ is typically set to be -1

Point $\mathbf{x}_\alpha^{(k)}$ lies on the straight line joining points $\overline{\mathbf{x}}^{(k)}$ and $\mathbf{x}_M^{(k)}$

- It is on the side of $\overline{\mathbf{x}}^{(k)}$, far from $\mathbf{x}_M^{(k)}$

Nelder and Mead (cont.)



$n = 2$, the centroid is midpoint of edge of $S^{(k)}$ opposite to $\mathbf{x}_M^{(k)}$

Nelder and Mead (cont.)

We fourthly compare $f\big[\mathbf{x}_\alpha^{(k)}\big]$ with f at the other vertices of the simplex

- Before accepting $\mathbf{x}_\alpha^{(k)}$ as the new vertex

We also try to move $\mathbf{x}_\alpha^{(k)}$ on the straight line joining $\overline{\mathbf{x}}^{(k)}$ and $\mathbf{x}_M^{(k)}$

To set the new simplex $S^{(k+1)}$

(1) If $f\big[\mathbf{x}_\alpha^{(k)}\big] < f\big[\mathbf{x}_m^{(k)}\big]$ (reflection produced a minimum), then

$$\mathbf{x}_\gamma^{(k)} = (1-\gamma)\overline{\mathbf{x}}^{(k)} + \gamma \mathbf{x}_M^{(k)}, \quad \text{with } \gamma < -1^4 \tag{50}$$

- Then, if $f\big[\mathbf{x}_\gamma^{(k)}\big] < f\big[\mathbf{x}_m^{(k)}\big]$, replace $\mathbf{x}_M$ by $\mathbf{x}_\gamma^{(k)}$
- Otherwise, $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\alpha^{(k)}$

We then proceed by incrementing k by one

[4] Typically, $\gamma = -2$

Nelder and Mead (cont.)

(2) If $f\big[\mathbf{x}_m^{(k)}\big] \leq f\big[\mathbf{x}_\alpha^{(k)}\big] < f\big[\mathbf{x}_\mu^{(k)}\big]$, then $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\alpha^{(k)}$

k is incremented by one

(3) If $f\big[\mathbf{x}_\mu^{(k)}\big] \leq f\big[\mathbf{x}_\alpha^{(k)}\big] < f\big[\mathbf{x}_M^{(k)}\big]$, we compute

$$\mathbf{x}_\beta^{(k)} = (1-\beta)\overline{\mathbf{x}}^{(k)} + \beta \mathbf{x}_\alpha^{(k)}, \quad \text{with } \beta > 0^5 \tag{51}$$

- Then, if $f\big[\mathbf{x}_\beta^{(k)}\big] > f\big[\mathbf{x}_M^{(k)}\big]$ define the vertices $S^{(k+1)}$

$$\mathbf{x}_i^{(k+1)} = \frac{1}{2}\big[\overline{\mathbf{x}}^{(k)} + \mathbf{x}_m^{(k)}\big] \tag{52}$$

- Otherwise $\mathbf{x}_M^{(k)}$ is replaced by $\mathbf{x}_\beta$

Then, we increment k

[5] with typically $\beta = 1/2$

Nelder and Mead (cont.)

(4) If $f[\mathbf{x}_\alpha^{(k)}] > f[\mathbf{x}_M^{(k)}]$, we compute

$$\mathbf{x}_\beta = (1-\beta)\overline{\mathbf{x}}^{(k)} + \beta \mathbf{x}_M^{(k)}, \quad \text{with } \beta > 0 \tag{53}$$

- If $f[\mathbf{x}_\beta^{(k)}] > f[\mathbf{x}_M^{(k)}]$ define the vertices of $S^{(k+1)}$

$$\mathbf{x}_i^{(k+1)} = \frac{1}{2}[\overline{\mathbf{x}}^{(k)} + \mathbf{x}_m^{(k)}]$$

- Otherwise we replace $\mathbf{x}_M^{(k)}$ with $\mathbf{x}_\beta^{(k)}$

Then we increment k

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

When the stopping criterion $\max_{i=0,\dots,n} ||\mathbf{x}_i^{(k)} - \mathbf{x}_m^{(k)}||_\infty < \varepsilon$ is met

$\rightsquigarrow$ $\mathbf{x}_m^{(k)}$ is retained as approximation of the minimiser

Convergence is guaranteed in very special cases only

Stagnation may occur, algorithm needs to be restarted

- The algorithm is nevertheless quite robust
- It is efficient for small dimensional problems
- Convergence rate depends on initial simplex

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)

Example

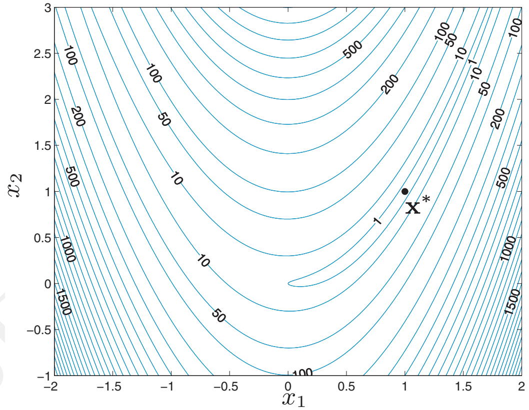
The Rosenbrock function

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

The global minimum is at $\mathbf{x}^* = (1, 1)$, and variation around $\mathbf{x}^*$ is low

Unconstrained optimisation
UFC/DC
CK0031/CK0248
2018.2
Unconstrained optimisation
Newton method
Line-search
Descent directions
Step-length α_k
Newton directions
Quasi-Newton directions
Gradient and conjugate-gradient directions
Trust-region
Nonlinear least-squares
Gauss-Newton
Levenberg-Marquardt
Derivative-free
Golden section and quadratic interpolation
Nelder and Mead

Nelder and Mead (cont.)



Nelder and Mead (cont.)

The simplex method

The M-command is `fminsearch`

```
x_0 = [-1.2,+1.0];

fun = @(x) (1-x(1))^2 + 100*(x(2)-x(1)^2)^2;

xstar = fminsearch(fun,x_0)

xstar =
1.000022021783570  1.000042219751772
```

To obtain additional information on the minimum value of f

```

[xstar,fval,exitflag,output] = fminsearch(fun,x_0)
```