Aalto University

# Root-finding with Newton-type methods

## CHEM-E7225 (was E7195), 2024

**Francesco Corona** (¬_¬)

Chemical and Metallurgical Engineering
School of Chemical Engineering

# Overview

Some notions in mathematical and numerical analysis that are used in optimisation

- Only instrumental concepts, to solve optimal control problems

Optimisation refers to the problem of finding the value of the inputs (independent variables) to some function such that the corresponding outputs (dependent variables) take an optimal value, where optimality is defined in some sense by the function itself

- $\rightsquigarrow$ The task can be formulated as a root-finding problem
- $\rightsquigarrow$ (The problem of finding the zeros of a function)

We will focus to a specific class of solution approaches known as Newton-type methods

# Preliminaries

## Root-finding with Newton-type methods

# Preliminaries

Let function $f$ be a twice-differentiable function (first and second derivatives) on $\mathcal{R}^N$

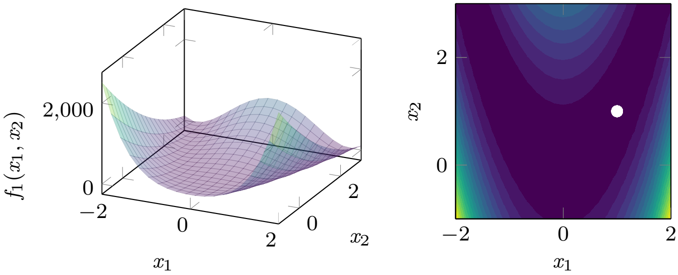
$$f : \mathcal{R}^N \rightarrow \mathcal{R}, \quad f \in \mathcal{C}^2(\mathcal{R}^N)$$

We shall use function $f$ to refresh some basic notions from multivariate calculus

- We are mainly interested in its gradient vector and its Hessian matrix

---

We consider a **Rosenbrock's function**, a classic benchmark for optimisation methods

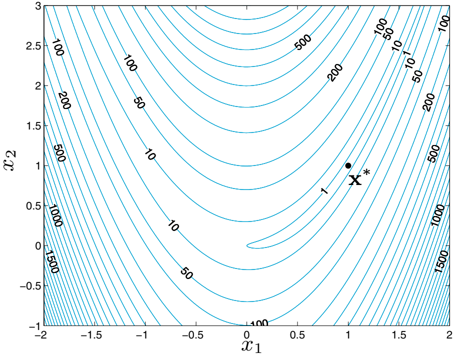



$$f : \mathcal{R}^2 \rightarrow \mathcal{R}$$

# Preliminaries | A scalar function

## Example

The **Rosenbrock's function**




$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Function $f(x)$ has a global minimum

$$x^* = (1, 1)$$

# Preliminaries | Gradient

Let the symbol $\nabla f(x)$ denote the **gradient vector** of function $f$ at some point $x \in \mathcal{R}^N$

$$\nabla f(x_1, x_2, \ldots, x_N) = \underbrace{\begin{bmatrix} \dfrac{\partial f(x_1, x_2, \ldots, x_N)}{\partial x_1} \\ \dfrac{\partial f(x_1, x_2, \ldots, x_N)}{\partial x_2} \\ \vdots \\ \dfrac{\partial f(x_1, x_2, \ldots, x_N)}{\partial x_N} \end{bmatrix}}_{N \times 1}$$

At any point $x \in \mathcal{R}^N$, we can define a vector of first derivatives, the gradient of $f$ at $x$

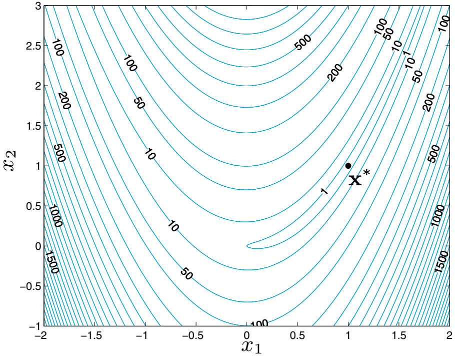
$$\nabla f(x) = \begin{bmatrix} \partial f(x)/\partial x_1 & \partial f(x)/\partial x_2 & \cdots & \partial f(x)/\partial x_N \end{bmatrix}_x^T$$

The symbol $\nabla = \begin{bmatrix} \partial/\partial x_1 & \partial/\partial x_2 & \cdots & \partial/\partial x_N \end{bmatrix}^T$ denotes the **gradient operator**

# Preliminaries | Gradient (cont.)

## Example

The **Rosenbrock's function**



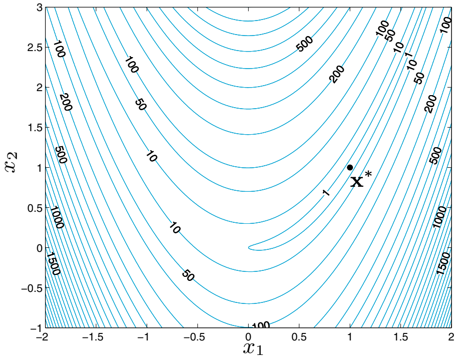

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$\frac{\partial f(x)}{\partial x_1} = -400x_1(x_2 - x_1^2) - 2(1 - x_1)$$

$$\frac{\partial f(x)}{\partial x_2} = 200(x_2 - x_1^2)$$

$$\nabla f(x) = \begin{bmatrix} \dfrac{\partial f(x_1, x_2)}{\partial x_1} \\[2ex] \dfrac{\partial f(x_1, x_2)}{\partial x_2} \end{bmatrix}$$

# Preliminaries | Gradient (cont.)




$$\nabla f\ (x) = \begin{bmatrix} \dfrac{\partial f\ (x)}{\partial x_1} \\[2ex] \dfrac{\partial f\ (x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}$$

Consider some point $x'$, say $x' = (0, 0)$, we can evaluate the gradient vector of $f$ at $x'$

$$\nabla f\ (x') = \begin{bmatrix} -400{x'}_1({x'}_2 - {x'}_1^2) - 2(1 - {x'}_1) \\ 200({x'}_2 - {x'}_1^2) \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

# Preliminaries | Hessian

Let the symbol $\nabla^2 f(x)$ denote the **Hessian matrix** of function $f$ at point $x \in \mathcal{R}^N$

$$\nabla^2 f(x) = \underbrace{\begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_N} \\ \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(x)}{\partial x_N \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_N \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_N \partial x_N} \end{bmatrix}}_{N \times N}$$

At point $x \in \mathcal{R}^N$, we can define a matrix of second derivatives, the Hessian of $f$ at $x$

$$\nabla^2 f(x) = \left[h_{ij}\right]_{\substack{i=1 \\ j=1}}^{N},$$

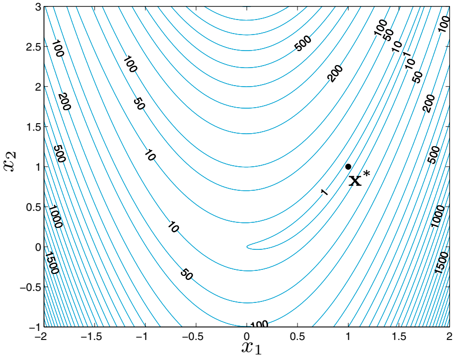
with $h_{ij} = \dfrac{\partial^2 f(x_1, \dots, x_i, \dots, x_j, \dots, x_N)}{\partial x_j \partial x_i}$, a symmetric $(N \times N)$ matrix

# Preliminaries | Hessian (cont.)

## Example

The **Rosenbrock's function**

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$




$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\[2ex] \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_2} \end{bmatrix}$$

$$\frac{\partial^2 f(x)}{\partial x_1 \partial x_1} = 1200 x_1^2 - 400 x_2 + 2$$

$$\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} = -400 x_1$$

$$\frac{\partial^2 f(x)}{\partial x_2 \partial x_1} = -400 x_1$$

$$\frac{\partial^2 f(x)}{\partial x_2 \partial x_2} = 200$$

# Preliminaries | Hessian (cont.)

Function $f(x)$,

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

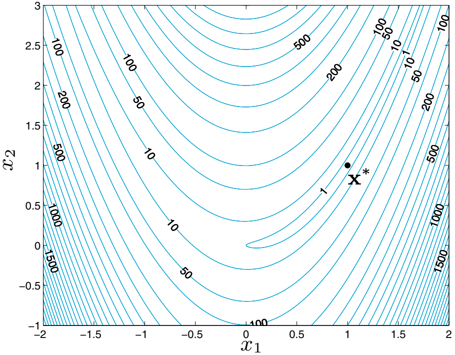
Gradient vector $\nabla f(x)$,

$$\begin{aligned}
\nabla f(x) &= \begin{bmatrix} \dfrac{\partial f(x)}{\partial x_1} & \dfrac{\partial f(x)}{\partial x_2} \end{bmatrix}_x^T \\
&= \begin{bmatrix} -400x_1(x_2 - x_1^2) - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix}
\end{aligned}$$

Hessian matrix $\nabla^2 f(x)$,

$$\begin{aligned}
\nabla^2 f(x) &= \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\[2ex] \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_2} \end{bmatrix} \\
&= \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}
\end{aligned}$$

# Preliminaries | Hessian (cont.)




$$\nabla^2 f\left(x\right) = \begin{bmatrix} \dfrac{\partial^2 f\left(x\right)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f\left(x\right)}{\partial x_1 \partial x_2} \\[2ex] \dfrac{\partial^2 f\left(x\right)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f\left(x\right)}{\partial x_2 \partial x_2} \end{bmatrix}$$

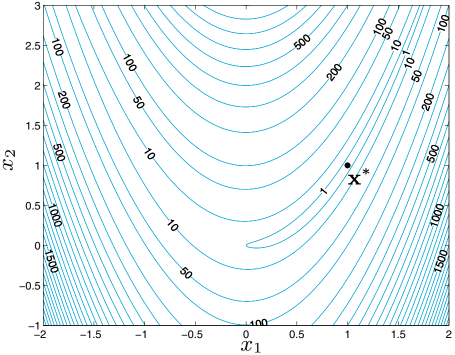
$$= \begin{bmatrix} 1200 x_1^2 - 400 x_2 + 2 & -400 x_1 \\ -400 x_1 & 200 \end{bmatrix}$$

Consider some point $x'$, say $x' = (0, 0)$, we can evaluate the Hessian matrix of $f$ at $x'$

$$\nabla^2 f\left(x'\right) = \begin{bmatrix} 1200 {x'}_1^2 - 400 {x'}_2 + 2 & -400 {x'}_1 \\ -400 {x'}_1 & 200 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 200 \end{bmatrix}$$

# Preliminaries | Hessian (cont.)




$$\nabla^2 f(x) = \begin{bmatrix} \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_2} \end{bmatrix}$$

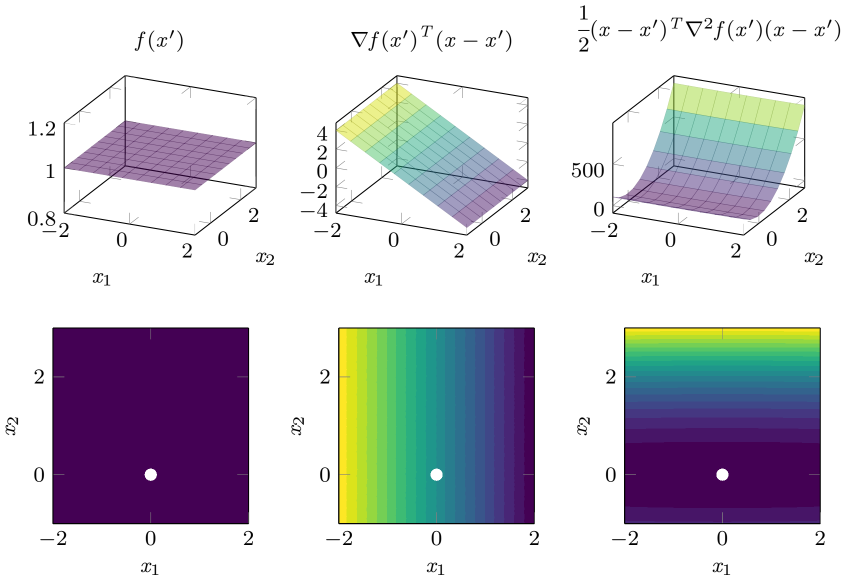
$$= \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}$$

Consider some point $x'$, say $x' = (1,1)$, we can evaluate the Hessian matrix of $f$ at $x'$

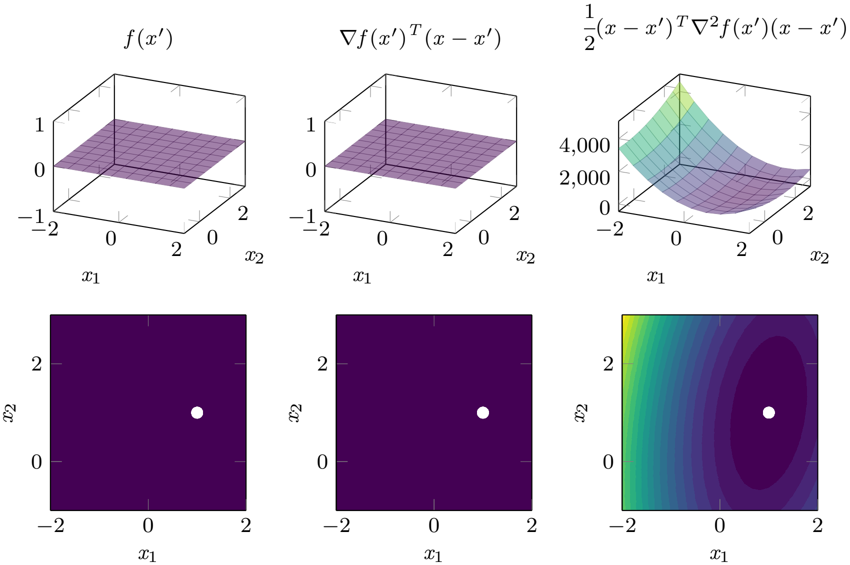
$$\nabla^2 f(x') = \begin{bmatrix} 1200{x'}_1^2 - 400{x'}_2 + 2 & -400{x'}_1 \\ -400{x'}_1 & 200 \end{bmatrix}$$

$$= \begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$$

# Preliminaries | Taylor's expansion



# Preliminaries | Taylor's expansjion(cont.)



# The Newton method

## Root-finding with Newton-type methods

# The Newton method

Let function $g : \mathcal{R}^N \to \mathcal{R}$ with $N \geq 1$ be some function belonging to class $\mathcal{C}^2(\mathcal{R}^N)$

- We are interested in solving the system of nonlinear equations $\nabla g(x) = 0$

That is, we are interested in the point(s) $x^* = (x_1^*, x_2^*, \dots, x_N^*)$ such that

$$\nabla g(x^*) = \begin{bmatrix} \dfrac{\partial g(x_1, x_2, \dots, x_N)}{\partial x_1} \\ \dfrac{\partial g(x_1, x_2, \dots, x_N)}{\partial x_2} \\ \vdots \\ \dfrac{\partial g(x_1, x_2, \dots, x_N)}{\partial x_N} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Points in which all the partial derivatives of $g$ are zero are **stationary points**

- We want to know where these fixed points or extrema are located

# The Newton method (cont.)

$$\nabla g\left(x^*\right)=\begin{bmatrix}\dfrac{\partial g\left(x_1,x_2,\ldots,x_N\right)}{\partial x_1}\\\dfrac{\partial g\left(x_1,x_2,\ldots,x_N\right)}{\partial x_2}\\\vdots\\\dfrac{\partial g\left(x_1,x_2,\ldots,x_N\right)}{\partial x_N}\end{bmatrix}=\begin{bmatrix}f_1\left(x_1,x_2,\cdots,x_N\right)\\f_2\left(x_1,x_2,\cdots,x_N\right)\\\vdots\\f_N\left(x_1,x_2,\cdots,x_N\right)\end{bmatrix}=\begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$$

Points $x^*$ are points in which all the functions $\{f_1, f_2, \ldots, f_N\}$ are jointly equal to zero

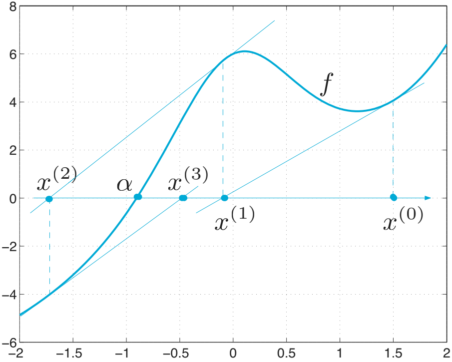
# The Newton method | Baby Newton

## Tangent's method (Baby Newton)

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

$\rightsquigarrow$ We are interested in point(s) $\alpha \in [a, b]$ such that $f(\alpha) = 0$

We know the function corresponding to the tangent to $f(x)$ at any point $x^{(k)} \in [a, b]$



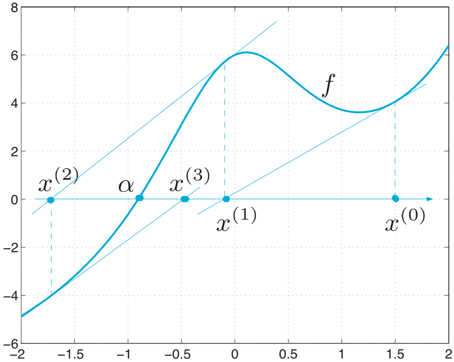

$$y(x) = f\left(x^{(k)}\right) + \underbrace{f'\left(x^{(k)}\right)}_{\left.\frac{df(x)}{dx}\right|_{x^{(k)}}} \left(x - x^{(k)}\right)$$

$\rightsquigarrow$ $k = 0$, the tangent to $f$ at $x^{(0)}$

$$y(x) = f\left(x^{(0)}\right) + f'\left(x^{(0)}\right)\left(x - x^{(0)}\right)$$

# The Newton method | Baby Newton (cont.)

We can use the tangent to find the point $x = x^{(k+1)}$, the point such that $y\left(x^{(k+1)}\right) = 0$




$$0 = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right)$$

$$\rightsquigarrow \quad x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$$

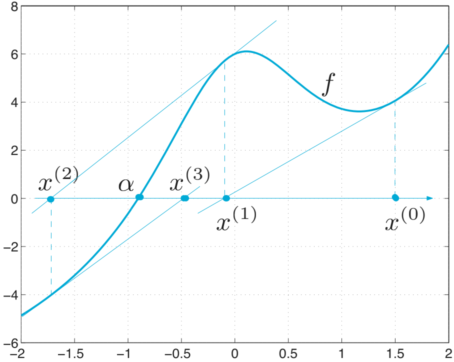
$\rightsquigarrow$ From $x^{(0)}$, we solve for point $x^{(1)}$

$$x^{(1)} = x^{(0)} - \frac{f\left(x^{(0)}\right)}{f'\left(x^{(0)}\right)}$$

# The Newton method | Baby Newton (cont.)

The operation can be repeated for $k = 0, 1, 2, \dots$, to find points $x^{(k+1)}$ approaching $\alpha$

- As $f\left(x^k\right) \to 0$ also differences $|x^{(k+1)} - x^{(k)}| \to 0$ (asymptotically, as $x^{(k)} \to \alpha$)




$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$$

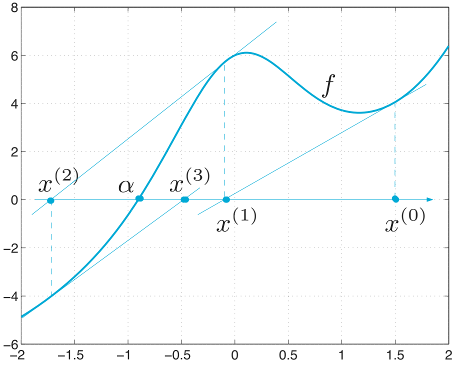
$\rightsquigarrow$ Importantly, the derivative $f'\left(x^{(k)}\right)$ must exists and must be non-zero

# The Newton method | Baby Newton (cont.)

$$y(x) = f\left(x^{(k)}\right) + f'\left(x^{(k)}\right)\left(x - x^{(k)}\right)$$

$$x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$$

The recursion defines the sequence $\{x^{(k)}\}$ that is generated by the **Newton's method**




- The method reduces to locally substituting $f$ with its tangent
- The substitution is repeated, until **convergence**

# The Newton method | Convergence tests

Newton's method returns the exact value $\alpha$ after an infinite number of iterations

- (Whenever it converges!)

In general, we would be content to obtain an approximation which is $\varepsilon$-accurate

- We would end the recursion when the user-specified tolerance $\varepsilon$ is reached

$$\underbrace{\|\alpha - x^{(k)}\|}_{e^{(k)}} < \varepsilon$$

To perform the test, we would need to know the exact value of $\alpha$ (the unknown)

In practice, an estimator of this error based on measurable quantities is used

- For the Newton's method, this could be the difference between iterates

$$\underbrace{\|x^{(k)} - x^{(k-1)}\|}_{\widehat{e}^{(k)}} < \varepsilon$$

- Alternatively, there is also the possibility to use the residuals

$$\| \underbrace{f\left(x^{(k)}\right)}_{f\left(x^{(k)}\right)-0} \| < \varepsilon$$

# The Newton method (cont.)

Consider the set of nonlinear equations from the vector-valued function $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$

$$
\begin{aligned}
f_1\left(x_1, x_2, \ldots, x_N\right) &= 0 \\
f_2\left(x_1, x_2, \ldots, x_N\right) &= 0 \\
&\vdots \\
f_N\left(x_1, x_2, \ldots, x_N\right) &= 0
\end{aligned}
$$

As we are letting $f = (f_1, \ldots, f_N)^T$ and $x = (x_1, \ldots, x_N)^T$, we can re-write the system

$$f\left(x\right) = 0$$

---

We are interested in solving this system of equations, by extending Newton's method

$$
\begin{aligned}
x^{(k+1)} &= x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} \\
&= x^{(k)} - \left(f'\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right)
\end{aligned}
$$

For $N > 1$, we firstly need to replace the first derivative with a set of first derivatives

# The Newton method | The Jacobian

$$
\begin{aligned}
f_1(x_1, x_2, \dots, x_N) &= 0 \\
f_2(x_1, x_2, \dots, x_N) &= 0 \\
&\vdots \\
f_N(x_1, x_2, \dots, x_N) &= 0
\end{aligned}
$$

We have $N$ functions, each in $N$ variables, we collect all the derivatives of function $f$

Let the symbol $J_f(x)$ denote the **Jacobian matrix** of function $f$

$$
J_f(x) = \underbrace{\begin{bmatrix}
\dfrac{\partial f_1(x)}{\partial x_1} & \dfrac{\partial f_1(x)}{\partial x_2} & \cdots & \dfrac{\partial f_1(x)}{\partial x_N} \\
\dfrac{\partial f_2(x)}{\partial x_1} & \dfrac{\partial f_2(x)}{\partial x_2} & \cdots & \dfrac{\partial f_2(x)}{\partial x_N} \\
\vdots & \vdots & \ddots & \vdots \\
\dfrac{\partial f_N(x)}{\partial x_1} & \dfrac{\partial f_N(x)}{\partial x_2} & \cdots & \dfrac{\partial f_N(x)}{\partial x_N}
\end{bmatrix}}_{N \times N}
$$

The Jacobian $J_f$ of function $f$ is the multivariate equivalent of the first derivarive $f'$

# The Newton method | The Jacobian (cont.)

Note how the $N$ rows of the Jacobian matrix of the vector-valued function $f$ correspond to the transpose of the $N$ gradient vectors of the (scalar) components of function $f$

$$J_f(x) = \underbrace{\begin{bmatrix} \dfrac{\partial f_1(x)}{\partial x_1} & \dfrac{\partial f_1(x)}{\partial x_2} & \cdots & \dfrac{\partial f_1(x)}{\partial x_N} \\ \dfrac{\partial f_2(x)}{\partial x_1} & \dfrac{\partial f_2(x)}{\partial x_2} & \cdots & \dfrac{\partial f_2(x)}{\partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_N(x)}{\partial x_1} & \dfrac{\partial f_N(x)}{\partial x_2} & \cdots & \dfrac{\partial f_N(x)}{\partial x_N} \end{bmatrix}}_{N \times N}$$

$$= \begin{bmatrix} \nabla f_1(x)^T \\ \nabla f_2(x)^T \\ \vdots \\ \nabla f_N(x)^T \end{bmatrix}$$

# The Newton method (cont.)

Consider the recursion of the univariate Newton's method, as we derived it earlier

$$x^{(k+1)} = x^{(k)} - \left(f'\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right)$$

We can define $\delta x = x^{(k+1)} - x^{(k)}$ and understand the recursion as follows

$$\begin{aligned} \text{Solve for } \delta x^{(k)} \quad & f'\left(x^{(k)}\right) \delta x^{(k)} = -f\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} + \delta x^{(k)} \end{aligned}$$

---

For the general case, let $x^{(0)} \in \mathcal{R}^N$ be an initial solution then for $k = 0, 1, \ldots$

$$\begin{aligned} \text{Solve for } \delta x^{(k)} \quad & J_f\left(x^{(k)}\right) \delta x^{(k)} = -f\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} + \delta x^{(k)} \end{aligned}$$

The operation is repeated until convergence

- That is, until $\delta x$ is small enough

# The Newton method (cont.)

$$\text{Solve for } \delta x \quad \underbrace{J_f\left(x^{(k)}\right)}_{A}\underbrace{\delta x^{(k)}}_{(x)} = \underbrace{-f\left(x^{(k)}\right)}_{b}$$

$$\text{Compute} \quad x^{(k+1)} = x^{(k)} + \delta x^{(k)}$$

A system of linear equations with coefficient matrix $J_f\left(x^{(k)}\right)$ is solved at each iteration

$$\underbrace{\delta x^{(k)}}_{x} = -J_f^{-1}\left(x^{(k)}\right) f\left(x^{(k)}\right)$$

---

It is possible to re-write the Newton method as we derived it, as an iteration scheme

$$\text{Solve for } \delta x \quad J_f\left(x^{(k)}\right)\underbrace{\delta x^{(k)}}_{\left(x^{(k+1)}-x^{(k)}\right)} = -f\left(x^{(k)}\right)$$

$$\text{Compute} \quad x^{(k+1)} = x^{(k)} + \underbrace{\delta x^{(k)}}_{-J_f^{-1}\left(x^{(k)}\right)f\left(x^{(k)}\right)}$$

# The Newton method (cont.)

Starting at $k = 0$ from some initial solution $x^{(0)}$, we have the Newton's step

$$x^{(k+1)} = x^{(k)} - J_f^{-1}\left(x^{(k)}\right) f\left(x^{(k)}\right), \quad k = 0, 1, \dots$$

Each Newton step moves $x^{(k)}$ in the (opposite) direction of vector $f\left(x^{(k)}\right)$

- The direction is also rotated according to matrix $J_f^{-1}\left(x^{(k)}\right)$

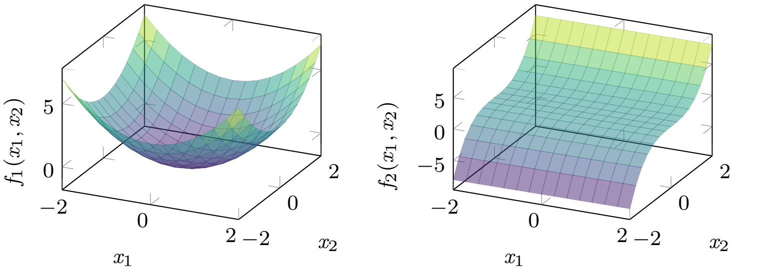
# The Newton method (cont.)

## Example

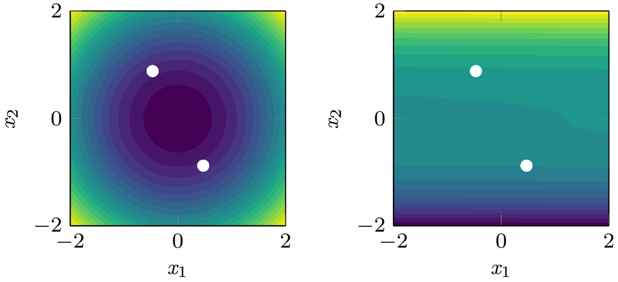
Consider the function $f(x) = (f_1(x_1, x_2), f_2(x_1, x_2))^T$

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

We are interested in point(s) $x^* = (x_1^*, x_2^*)$ where $f(x^*) = 0$



# The Newton method (cont.)




The problem has two solutions, two points where both function $f_1$ and $f_2$ equal to zero

- $\approx (0.47, -0.88)^T$
- $\approx (-0.47, 0.88)^T$

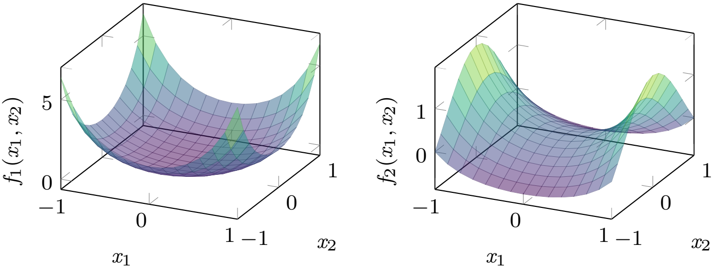
# The Newton method (cont.)

## Example

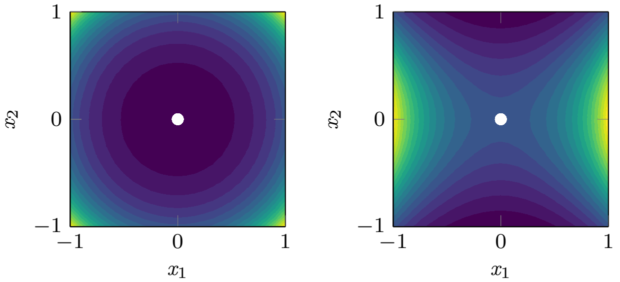
Consider the function $f(x) = \big(f_1(x_1, x_2), f_2(x_1, x_2)\big)^T$

$$\begin{cases} f_1(x_1, x_2) = e^{\left(x_1^2 + x_2^2\right)} - 1 \\ f_2(x_1, x_2) = e^{\left(x_1^2 - x_2^2\right)} - 1 \end{cases}$$

We are interested in point(s) $x^* = (x_1^*, x_2^*)$ where $f(x^*) = 0$



# The Newton method (cont.)




The problem has a unique solution, the point where both function $f_1$ and $f_2$ are zero

$\rightsquigarrow$ $(0,0)^T$

# The Newton method | Towards optimisation

Our main interest in root-finding is on certain vector-valued functions $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$

- Functions $f$ that are the gradient of twice-differentiable functions $g : \mathcal{R}^N \rightarrow \mathcal{R}$

$$f(x) = \nabla g(x)$$

We are interested in point(s) $x^* \in \mathcal{R}^N$ such that $\nabla g(x) = 0$

- Extrema: Minima, maxima, and saddle points of $g(x)$

---

$$\begin{aligned} \text{Solve for } \delta x \quad & J_f\left(x^{(k)}\right)\delta x^{(k)} = -f\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} + \delta x^{(k)} \end{aligned}$$

$\rightsquigarrow$ Function $f$ is gradient $\nabla g(x)$ of $g$, the Jacobian $J_f\left(x^{(k)}\right)$ is its Hessian $\nabla^2 g\left(x^k\right)$

$$\begin{aligned} \text{Solve for } \delta x \quad & \nabla^2 g\left(x^{(k)}\right)\delta x^{(k)} = -\nabla g\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} + \delta x^{(k)} \end{aligned}$$

# The Newton method | Towards optimisation (cont.)

$$\begin{aligned} \text{Solve for } \delta x \quad & \nabla^2 g\left(x^{(k)}\right) \delta x^{(k)} = -\nabla g\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} + \delta x^{(k)} \end{aligned}$$

We can re-write the Newton method as we derived it, as an explicit iteration scheme

$$\begin{aligned} \text{Solve for } \delta x \quad & \nabla^2 g\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = -\nabla g\left(x^{(k)}\right) \\ \text{Compute} \quad & x^{(k+1)} = x^{(k)} - \left(\nabla^2 g(x^{(k)})\right)^{-1} \nabla g(x^{(k)}) \end{aligned}$$

That is, starting from some initial solution (guess) $x^{(0)}$

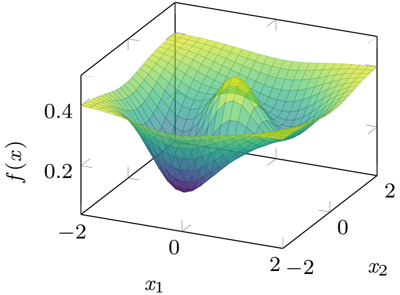
$$x^{(k+1)} = x^{(k)} - \left(\nabla^2 g\left(x^{(k)}\right)\right)^{-1} \nabla g\left(x^{(k)}\right), \quad k = 0, 1, \dots$$

# The Newton method | Towards optimisation (cont.)

## Example

Consider function $f : \mathcal{R}^2 \rightarrow \mathcal{R}$

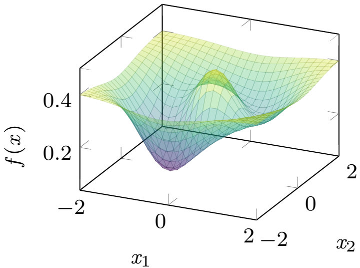
$$f(x) = \frac{2}{5} - \frac{1}{10}\left(5x_1^2 + 5x_2^2 + 3x_1x_2 - x_1 - 2x_2\right) e^{\left(-\left(x_1^2+x_2^2\right)\right)}$$

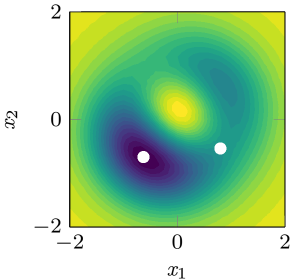



We are interested in those points $(x_1, x_2)$ where $\nabla f(x) = 0$

We can easily identify the points where the gradient vanishes

- Two minima
- A maximum
- A saddle




We can analyse solutions from the Netwon's method, from different initial points $x^{(0)}$

Suppose that we let $x^{(0)} = (-0.9, -0.9)$

$\rightsquigarrow$ After 5 iterations the method converges to $x = (-0.63058, -0.70074)$

Suppose that we let $x^{(0)} = (-1.0, -1.0)$

$\rightsquigarrow$ After 400 iterations the stopping criterion is still not fulfilled

Suppose that we let $x^{(0)} = (+0.5, -0.5)$

$\rightsquigarrow$ After 5 iterations the method converges to the saddle point

- $x = (0.80659, -0.54010)$

□

# The Newton method | Towards optimisation (cont.)

$$x^{(k+1)} = x^{(k)} - \left(J_f\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right)$$

$$x^{(k+1)} = x^{(k)} - \left(\nabla^2 g(x^{(k)})\right)^{-1} \nabla g\left(x^{(k)}\right)$$

In spite of a simple implementation, the Newton method is demanding for a large $N$

- The method requires analytic expressions of the derivatives
- The method also requires inverting the Jacobian (Hessian)
- Naive inversion of a $N \times N$ matrix is $\mathcal{O}(N^3)$

For the method to converge, it is also important that $x^{(0)}$ is chosen near enough $x^*$

# The Newton method | Towards optimisation (cont.)

$$x^{(k+1)} = x^{(k)} - \left(J_f\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right)$$

$$x^{(k+1)} = x^{(k)} - \left(\nabla^2 g(x^{(k)})\right)^{-1} \nabla g\left(x^{(k)}\right)$$

Flexibility is achieved by replacing Jacobians (Hessians) by invertible approximations

$$\underbrace{M^{(k)}}_{N \times N}$$

The use of invertible approximations leads the family of **Newton-type methods**

$$x^{(k+1)} = x^{(k)} - \left(\underbrace{M^{(k)}}_{(J_f(x^{(k)}))^{-1}}\right)^{-1} f\left(x^{(k)}\right)$$

$$x^{(k+1)} = x^{(k)} - \left(\underbrace{M^{(k)}}_{(\nabla^2 g(x^{(k)}))^{-1}}\right)^{-1} \nabla g\left(x^{(k)}\right)$$

# Newton-type methods

## Root-finding with Newton-type methods

# Newton-type methods

Consider some function $f \in \mathcal{C}^2(\mathcal{R}^N)$ bounded below, we are interested in its minima

- That is, we are interested in point $x^* \in \mathcal{R}^N$ such that $f(x^*)$ is the smallest
- The minima of $f$ occur at points $x$ where the gradient $\nabla f(x)$ is zero

---

We can use Newton and Newton-type recursions to find the zeros of function $\nabla f(x)$

- From some initial approximate solution $x^{(0)}$, we have

$$x^{(k+1)} = x^{(k)} + \underbrace{(-1)\left(M^{(k)}\right)^{-1}\nabla f(x^{(k)})}_{d^{(k)}}, \quad k = 0, 1 \dots$$

At iteration steps $k \geq 0$, let $x^{(k+1)}$ be the next point of the sequence

- Point $x^{(k+1)}$ depends on point $x^{(k)}$ and some vector $d^{(k)}$

The vector (its direction) $d^{(k)}$ depends on two terms

- $\rightsquigarrow$ The gradient vector $\nabla f\left(x^{(k)}\right)$ of $f$
- $\rightsquigarrow$ The Hesse matrix $\nabla^2 f\left(x^{(k)}\right)$ of $f$
- $\rightsquigarrow$ (Or, an approximation $M^{(k)}$)

# Newton-type methods (cont.)

$$x^{(k+1)} = x^{(k)} + (\underbrace{1}_{\alpha^{(k)}}) \underbrace{\left(M^{(k)}\right)^{-1} (-\nabla f(x^{(k)}))}_{d^{(k)}}, \quad k = 0, 1 \dots$$

We can also introduce a dependence on some parameter $\alpha_k \in \mathcal{R}_{\geq 0}$, the **step-length**

- In the basic implementation $a^{(k)}$ is constant with $k$ and equal to (plus) one
- $\nabla f \left(x^{(k)}\right)$ gives the direction of maximal positive growth of $f$ from $x^{(k)}$
- $\nabla^2 f \left(x^{(k)}\right)$ or $M^{(k)}$ applies a transformation to the gradient direction $f$

The negative sign sets the iterates to move downwards

# Newton-type methods (cont.)

The algorithmic formulation of a general Newton-type (line-search/descent) method

Let $x^{(0)} \in \mathcal{R}^N$ be an initial approximation of the minimiser

$$\begin{aligned} \text{Determine descent direction} \quad & d^{(k)} \in \mathcal{R}^N \\ \text{Compute step-length} \quad & \alpha^{(k)} \in \mathcal{R}_{\geq 0} \\ \text{Compute new approximation} \quad & x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)} \end{aligned}$$

# Newton-type methods (cont.)

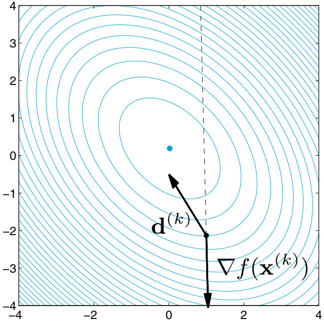
Because vector $d^{(k)}$ needs be a **descent direction**, it must satisfy certain conditions

$$\begin{cases} d^{(k)} \nabla f\left(x^{(k)}\right) < 0 & \left(\nabla f\left(x^{(k)}\right) \neq 0\right) \\ d^{(k)} = 0 & \left(\nabla f\left(x^{(k)}\right) = 0\right) \end{cases}$$

$\rightsquigarrow$ $d^{(k)} \nabla f\left(\mathbf{x}^{(k)}\right)$ is the directional derivative of $f$ along $d^{(k)}$

---

For example, consider a function $f$ (a quadratic form) and its gradient vector at $x^{(k)}$




$$\underbrace{(1)}_{\alpha^{(k)}} \underbrace{\left(M^{(k)}\right)^{-1} \left(-\nabla g(x^{(k)})\right)}_{d^{(k)}}$$

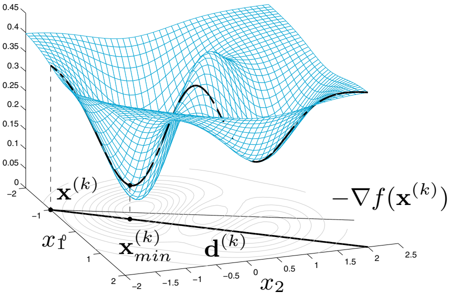
Direction $d^{(k)}$ must be a suitable descent direction

- Parameter $\alpha^{(k)}$ defines the step-size

# Newton-type methods (cont.)

The step-size $\alpha^{(k)}$ can be computed by solving a one-dimensional minimisation problem

- The minimisation of the restriction of function $f(x)$ along direction $d^{(k)}$
- The idea is to set $\alpha^{(k)}$ to reach $x^{(k)}_{\text{min}}$, the minimiser along $d^{(k)}$

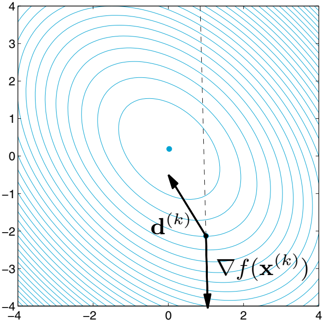



When $f$ or its restriction is not quadratic, the computation of $\alpha^{(k)}$ can be involved

- Certain (Wolfe's) conditions on $\alpha^{(k)}$ must be satisfied before it is accepted

# Newton-type methods | Step-lengths

Given a descent direction $d^{(k)}$, we would want the step-length $\alpha^{(k)}$ to be optimal




- Such that function $f$ is smallest along $d^{(k)}$
- (Along the restriction $f\left(x^{(k)} + \alpha^{(k)} d^{(k)}\right)$)

$$
\begin{aligned}
f\left(x^{(k)} + \alpha^{(k)} d^{(k)}\right) \approx f\left(x^{(k)}\right) & \\
+ \alpha^{(k)} \nabla^T f\left(x^{(k)}\right) d^{(k)} & \\
+ \frac{1}{2} (\alpha^{(k)})^2 d^{(k)} \nabla^2 f\left(x^{(k)}\right) d^{(k)} &
\end{aligned}
$$

By setting to zero the derivative with respect to $\alpha^{(k)}$ of a second-order approximation around $x^{(k)}$ of the restriction of function $f$ along the descent direction $d^{(k)}$, we get

$$
\alpha^{(k)} = \frac{-\nabla^T f\left(x^{(k)}\right) d^{(k)}}{d^{(k)} \nabla^2 f\left(x^{(k)}\right) d^{(k)}}
$$

# Newton-type methods | Newton directions

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \underbrace{\left(M^{(k)}\right)^{-1} (-\nabla f(x^{(k)}))}_{d^{(k)}}, \quad k = 0, 1 \dots$$

**Newton's directions**, $M^{(k)} = \nabla^2 f\left(x^{(k)}\right)$

$$d^{(k)} = -\left(\underbrace{\nabla^2 f\left(x^{(k)}\right)}_{M^{(k)} \succ 0}\right)^{-1} \nabla f\left(x^{(k)}\right)$$

# Newton-type methods | Newton directions (cont.)

$$d^{(k)} = -\Bigg(\underbrace{\nabla^2 f\left(x^{(k)}\right)}_{M^{(k)}}\Bigg)^{-1} \nabla f\left(x^{(k)}\right)$$

Consider functions $f$ such that the Hessian matrices $\{\nabla^2 f\left(x^{(k)}\right)\}$ are positive definite

- Also, suppose that for some $\kappa > 0$ the condition number of $\nabla^2 f\left(x^{(k)}\right)$

$$\begin{aligned}\mathcal{K}(\nabla^2 f\left(x^{(k)}\right)) &= \frac{\lambda_{\text{max}}(\nabla^2 f\left(x^{(k)}\right))}{\lambda_{\text{min}}(\nabla^2 f\left(x^{(k)}\right))}\\ &\leq \kappa \quad (\text{for all } k)\end{aligned}$$

Under these conditions, the sequence $\{x^{(k)}\}$ converges to a minimum $x^*$ of function $f$

# Newton-type methods | Newton directions (cont.)

The positive definiteness of $M^{(k)}$ must be over a large enough neighbourhood of $x^{(k)}$

If $M^{(k)} \succ 0$, then $d^{(k)} = -\left(M^{(k)}\right)^{-1} \nabla f\left(x^{(k)}\right)$ is a descent direction

$$d^{(k)} \nabla f\left(x^{(k)}\right) < 0 \quad \left(\nabla f\left(x^{(k)}\right) \neq 0\right)$$

We have,

$$\underbrace{d^{(k)} \nabla f\left(x^{(k)}\right) = -\underbrace{\nabla^T f\left(x^{(k)}\right) \underbrace{M^{(k)}}_{\succ 0} \nabla f\left(x^{(k)}\right)}_{>0}}_{<0}$$

---

Note that a descent direction do not necessarily imply a reduction in function value

- The step-length $\alpha^{(k)}$ may lead to $f\left(x^{(k+1)}\right) > f\left(x^{(k)}\right)$
- The step-length can be reduced, to avoid this risk

# Newton-type methods | Newton directions (cont.)

$$d^{(k)} = -\Big(\underbrace{\nabla^2 f\left(x^{(k)}\right)}_{M^{(k)}}\Big)^{-1} \nabla f\left(x^{(k)}\right)$$

For Hessians that are not positive definite, direction $d^{(k)}$ may not be a descent direction

$$d^{(k)} \nabla f\left(x^{(k)}\right) \geq 0 \quad \left(\nabla f\left(x^{(k)}\right) \neq 0\right)$$

- (Also Wolfe's conditions on the step-length may lose validity/meaning)

Against this, it is possible to add a diagonal or full matrix $E^{(k)}$ to the Hessian

$$\underbrace{\nabla^2 f\left(x^{(k)}\right) + E^{(k)}}_{M^{(k)}} \succ 0$$

# Newton-type methods | Quasi-Newton

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \underbrace{\left(M^{(k)}\right)^{-1} (-\nabla f(x^{(k)}))}_{d^{(k)}}, \quad k = 0, 1 \dots$$

**Quasi-Newton directions**, $M^{(k)} = \widetilde{\nabla}^2 f\left(x^{(k)}\right)$

$$d^{(k)} = -\left(\underbrace{\widetilde{\nabla}^2 f\left(x^{(k)}\right)}_{M^{(k)} \succ 0}\right)^{-1} \nabla f\left(x^{(k)}\right)$$

A common approach for constructing approximations of the Hessian matrix is **BFGS**

- The **Broyden**, **Fletcher**, **Galfarb**, and **Shanno**'s method

# Newton-type methods | Quasi-Newton (cont.)

Given an initial symmetric matrix $M^{(0)} \succ 0$, the BFGS method recursively computes

$$M^{(k+1)} = M^{(k)} + \frac{\left(\nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)\right)\left(\nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)\right)^T}{\left(\nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right)\right)^T \left(x^{(k+1)} - x^{(k)}\right)^T} - \frac{M^{(k)}\left(x^{(k+1)} - x^{(k)}\right)\left(x^{(k+1)} - x^{(x)}\right)^T M^{(k)}}{\left(x^{(k+1)} - x^{(k)}\right)^T M^{(k)} \left(x^{(k+1)} - x^{(k)}\right)}$$

---

The matrices from rank-one updates, as BFGS, need be symmetric and positive definite

This is guaranteed by the following condition,

$$(x^{(k+1)} - x^{(k)})^T(\nabla f\left(x^{(k)}\right) - \nabla f\left(x^{(k+1)}\right)) > 0$$

From a quadratic approximation of $f$ about $x^{(k)}$

# Newton-type methods | Quasi-Newton (cont.)

Given an approximate solution $x^{(0)}$ and a positive definite approximate Hessian $M^{(0)}$

We have the general formulation of the quasi-Newton's method

$$
\begin{aligned}
\text{Solve for } d^{(k)} \quad & M^{(k)} d^{(k)} = -\nabla f\left(x^{(k)}\right) \\
(\text{Compute } \alpha^{(k)} \quad & \text{Verify Wolfe's conditions}) \\
\text{Set } x^{(k+1)} \quad & x^{(k+1)} = x^{(k)} + \alpha^{(k)} d^{(k)} \\
\text{Compute} \quad & x^{(k+1)} - x^{(k)} \\
\text{Compute} \quad & \nabla f\left(x^{(k+1)}\right) - \nabla f\left(x^{(k)}\right) \\
\text{Compute} \quad & M^{(k+1)}
\end{aligned}
$$

# Newton-type methods | Gradient directions (cont.)

$$x^{(k+1)} = x^{(k)} \underbrace{- \left(M^{(k)}\right)^{-1} \nabla f(x^{(k)})}_{d^{(k)}}, \quad k = 0, 1 \dots$$

**Gradient directions (gradient descent, steppest descent, ...)**, $M^{(k)} = I$

$$d^{(k)} = - \left( \underbrace{I}_{M^{(k)}} \right)^{-1} \nabla f \left( x^{(k)} \right)$$

This approach is successfully utilised for large-scale optimisation problems

- Where large Hessian matrices are expensive to invert

# Newton-type methods | Gradient directions (cont.)

## Conjugate-gradient directions

$$d^{(0)} = -\nabla f\left(x^{(0)}\right)$$
$$d^{(k+1)} = -\nabla f\left(x^{(k+1)}\right) + \beta^{(k)} d^{(k)}, \quad k = 0, 1, \dots$$

There exist alternatives for computing parameter $\beta^{(k)}$, some commonly used ones

- **Fletcher-Reeves**

$$\beta_{\text{FR}}^{(k)} = \frac{\|\nabla f\left(x^{(k)}\right)\|^2}{\|\nabla f\left(x^{(k-1)}\right)\|^2}$$

- **Hestenes-Stiefel**

$$\beta_{\text{HS}}^{(k)} = \frac{\nabla f\left(x^{(k)}\right)^T \left(\nabla f\left(x^{(k)}\right) - \nabla f\left(x^{(k-1)}\right)\right)}{d^{(k-1)^T} \left(\nabla f\left(x^{(k)}\right) - \nabla f\left(x^{(k-1)}\right)\right)}$$

- **Polak-Ribiére**

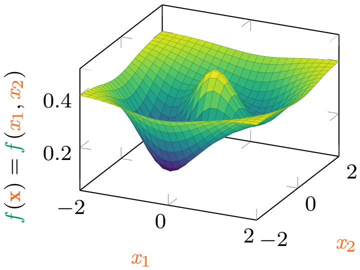
$$\beta_{\text{PR}}^{(k)} = \frac{\nabla f\left(x^{(k)}\right)^T \left(\nabla f\left(x^{(k)}\right) - \nabla f\left(x^{k-1}\right)\right)}{\|\nabla f\left(x^{(k-1)}\right)\|^2}$$

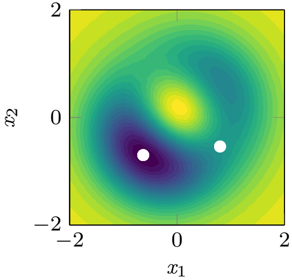
# Newton-type methods (cont.)

## Example

Consider function $f : \mathcal{R}^2 \rightarrow \mathcal{R}$

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

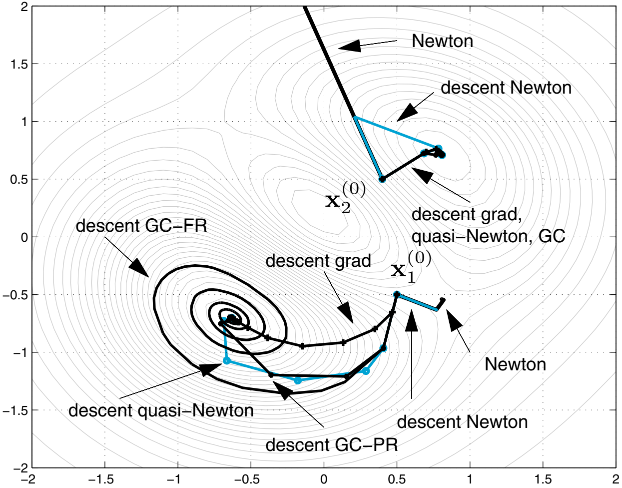



Compare sequences $\{x^{(k)}\}$ with Newton and quasi-Newton direction, from various $x^{(0)}$

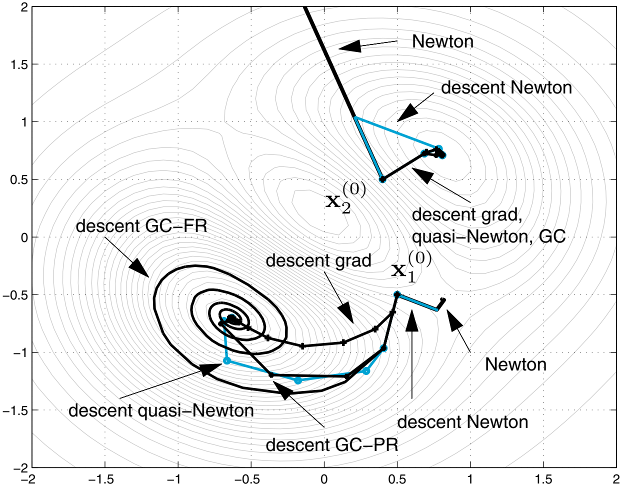
# Newton-type methods (cont.)

$x_1^{(0)} = (0.5, -0.5)$, Newton converges (to a saddle) and some inexact methods collapse



# Newton-type methods (cont.)

$x_2^{(0)} = (0.4, 0.5)$, Newton diverges but starts well together with some inexact methods



# Convergence rates

## Root-finding with Newton-type methods

# Convergence

Consider the set of nonlinear equations from the vector-valued function $f : \mathcal{R}^N \rightarrow \mathcal{R}^N$

$$f(x) = 0$$

We are interested in solving this system of equations, a root-finding problem

- That is, find $x^*$ such that $f(x^*) = 0$

The exact Newton method solves

$$f\left(x^{(k)}\right) + J_f\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = 0$$

We get the exact iterates,

$$x^{(k+1)} = x^{(k)} - \left(J_f\left(x^{(k)}\right)\right)^{-1} f\left(x^{(k)}\right)$$

The Newton-type iterates,

$$x^{(k+1)} = x^{(k)} - \left(M^{(k)}\right)^{-1} f\left(x^{(k)}\right)$$

$M^{(k)}$ must be an invertible and positive definite approximation of the Jacobian $J_f\left(x^{(k)}\right)$

# Convergence (cont.)

Let $x^{(k)}$ be the approximated solution at iteration $k$ and let $x^*$ denote the solution

Consider converging sequences of iterates $\{x^{(k)}\}$,

$$\lim_{k \to \infty} x^{(k)} = x^*$$

Or, equivalently,

$$\lim_{k \to \infty} \|x^{(k)} - x^*\| = 0$$

We are interested in characterising the rate at which iterates the $x^{(k)}$ converge to $x^*$

Consider the convergence condition,

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^p} \leq C \qquad (\text{for some } C \in (0,1) \text{ and } k \geq k_0)$$

**Order of convergence** is denoted by $p$ and $C$ is known as the **convergence factor**

- The condition is defined for an error, which is based on unknown $x^*$

The necessary condition for convergence is that $x^{(k_0)}$ is chosen sufficiently close to $x^*$

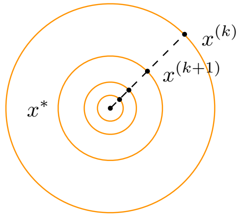
- Because of this, only **local convergence** properties can be established

# Convergence (cont.)

We define the following **(local) convergence rates** for the sequence $\{x^{(k)}\}$ of iterates

- **Q-linear**, for some $C^{(k)} \in (0,1)$ and for all $k = 0, 1, \ldots$

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq C^{(k)}$$

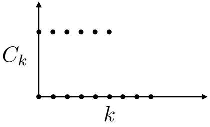



$C^{(k)}$ is the rate of convergence

- It remains constant with $k$

An equivalent form

$$\limsup_{k\to\infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} < 1$$



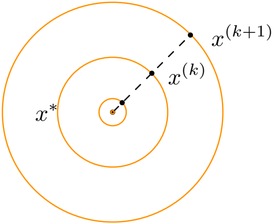

Linear contraction rates characterise an exponential decay of the approximation error

- An exponential decay (or growth) is not necessarily rapid, it depends on $C$

# Convergence (cont.)

- **Q-superlinear**, for some stable sequence $C^{(k)} \rightarrow 0$

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq C^{(k)}$$

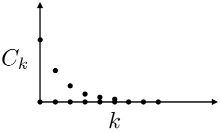



$C^{(k)}$ is the rate of convergence

- It shrinks with $k$

In the limit form,

$$\limsup_{k \rightarrow \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0$$




The rate of the exponential decay is not constant, but decays with the iteration count

- It is equivalent to an always increasing linear contraction rate

# Convergence (cont.)

- **Q-quadratic**, for some $C^{(k)} < \infty$ and for all $k = 0, 1, \dots$

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^2} \leq C^{(k)}$$

Rearranging terms, we have

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq \underbrace{C^{(k)}\|x^{(k)} - x^*\|}_{C^{(k)}(x^{(k)})}$$

$C^{(k)}(x^{(k)})$ is a local rate of convergence

- It shrinks with $k$ and $x^{(k)}$