Advanced Numerical Analysis

Lecture Notes

Prof. Daniel Kressner
EPFL / SB / MATHICSE / ANCHP

Spring 2015

May 21, 2015

Remarks

- Although not necessary, you may find it helpful to complement these lecture notes with additional literature. These lecture notes are based on material from

  [**DB**] P. Deuflhard and F. Bornemann. *Scientific computing with ordinary differential equations.* Texts in Applied Mathematics, 42. Springer, 2002.

  [**E**] F. Eisenbrand. Lecture notes on Discrete Optimization, 2013. Available from `http://disopt.epfl.ch/Opt2013`.

  [**HLW**] E. Hairer, Ch. Lubich, and G. Wanner. *Geometric numerical integration.* Structure-preserving algorithms for ordinary differential equations. Springer Series in Computational Mathematics, 31. Springer, 2010.

  [**HNW**] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I.* Nonstiff problems. Second edition. Springer Series in Computational Mathematics, 8. Springer, 1993.

  [**HW**] E. Hairer and G. Wanner. *Solving ordinary differential equations. II.* Stiff and differential-algebraic problems. Second revised edition. Springer Series in Computational Mathematics, 14. Springer-Verlag, Berlin, 2010.

  [**K**] C. T. Kelley. *Iterative Methods for Optimization.* SIAM, 1999.

  [**Kr**] J. Krieger. *Equations Différentielles Ordinaires*, 2014. Lecture notes available from `http://pde.epfl.ch/page-44573-en.html`.

  [**N**] Yu. Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course*, 2004.

  [**NW**] J. Nocedal and S. J. Wright. *Numerical optimization.* Second edition. Springer Series in Operations Research and Financial Engineering. Springer, 2006.

  [**UU**] M. Ulbrich and S. Ulbrich. *Nichtlineare Optimierung*, Birkhäuser Verlag, 2012.

  Please note: None of these books is needed to prepare for the exam. These lecture notes are entirely sufficient.

- Sections or paragraphs marked by a $\star$ contain complementary material that is *not* part of the exam. You may skip reading these sections, but it may be worthwhile not to do so.

- If you have any suggestions or spot mistakes, please send a message to

`daniel.kressner@epfl.ch`
or `michael.steinlechner@epfl.ch`

Contents

Part I

Ordinary Differential Equations

Chapter 1

Introduction and Review

The first part of this course starts where the course *Analyse Numérique* ended: the numerical solution of ordinary differential equations (ODEs). In this first chapter, we will therefore very briefly review (and slightly extend) some of the material covered in *Analyse Numérique*.

In general, an **initial value problem** *[problème de Cauchy]* takes the form

$$\begin{cases} \mathbf{y}'(t) = f\big(t, \mathbf{y}(t)\big) & t \in I, \\ \mathbf{y}(t_0) = \mathbf{y}_0, \end{cases} \tag{1.1}$$

where:

- $t \in \mathbb{R}$ denotes **time**.
- I is a real interval containing the initial time t_0.
- $\mathbf{y}(t) \in D \subset \mathbb{R}^d$ is the desired solution vector at time t. It contains d time-dependent functions $y_1(t), \ldots, y_d(t)$. Often, $\mathbf{y}(t)$ is called the **state** *[état]* of the ODE. The open set D containing all admissible states is called the **phase space** *[espace des phases]* and $\Omega := I \times D \subset \mathbb{R}^{d+1}$ is called the **augmented phase space**.
- $\mathbf{y}'(t)$ is the entry-wise derivative of $\mathbf{y}$ with respect to time:

$$\mathbf{y}'(t) = \begin{pmatrix} y_1'(t) \\ \vdots \\ y_d'(t) \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial t} y_1(t) \\ \vdots \\ \frac{\partial}{\partial t} y_d(t) \end{pmatrix}.$$

- $f : \Omega \to \mathbb{R}^d$ is a continuous function depending on time and state. If there is no direct dependence on time (for example, $f(t, y(t)) = y(t)^2$) then the ODE is called **autonomous** *[équation différentielle autonome]* and we can write $f(\mathbf{y}(t))$ instead of $f(t, \mathbf{y}(t))$.

1.1 Existence and uniqueness of solutions

In the following, we briefly recall classical results on the existence of a solution $\mathbf{y} \in C^1(I, \mathbb{R}^d)$ to the initial value problem (1.1). See Lectures 2 and 3 of [Kr].

Theorem 1.1 (Peano existence theorem) *Consider an initial value problem* (1.1) *with* $f \in C^0(\Omega, \mathbb{R}^d)$ *and initial data* $(t_0, \mathbf{y}_0) \in \Omega$. *Then there is at least one solution* $\mathbf{y} \in C^1([t_0, t_0 + \alpha])$ *for some* $\alpha > 0$.

Uniqueness requires a condition on f that is stronger than continuity.

Definition 1.2 *Consider a mapping* $f \in C(\Omega, \mathbb{R})$ *with* $\Omega = I \times D \subset \mathbb{R} \times \mathbb{R}^d$. *Then* f *is called* **locally Lipschitz continuous** *[localement lipschitzienne] on* Ω *with respect to the state variable if for any compact subset* $K \subset \Omega$ *there exists a Lipschitz constant* L_K *such that*

$$\|f(t, \mathbf{y}) - f(t, \widetilde{\mathbf{y}})\| \leq L_K \|\mathbf{y} - \widetilde{\mathbf{y}}\| \qquad \forall (t, \mathbf{y}), (t, \widetilde{\mathbf{y}}) \in K.$$

In the definition above, $\|\cdot\|$ denotes an arbitrary vector norm.

Theorem 1.3 (Picard–Lindelöf theorem) *Consider an initial value problem* (1.1) *with* $f \in C^0(\Omega, \mathbb{R}^d)$ *locally Lipschitz continuous on* Ω *with respect to the state variable, and* $(t_0, x_0) \in \Omega$. *Then there exists* $\alpha > 0$ *such that the following holds: There exists one and only one solution* $\mathbf{y} \in C^1([t_0, t_0 + \alpha])$ *of the initial value problem on the interval* $[t_0, t_0 + \alpha]$.

Often, solutions can be continued beyond the interval $[t_0, t_0 + \alpha]$ guaranteed by Theorem 1.3. For harmless ODEs, this continuation may extend until $T = \infty$. If the extension of a solution $\mathbf{y}(t)$ breaks down at some point T then because of one of the following two reasons:

1. **Blow up** of the solution: $\lim\limits_{\substack{t \to T \\ t < T}} \|\mathbf{y}(t)\| = \infty$.
2. **Collapse** of the solution: $\lim\limits_{\substack{t \to T \\ t < T}} \operatorname{dist}\big((t, \mathbf{y}(t)), \partial\Omega\big) = 0$.

First, an example for blow up.

Example 1.4 Let $\Omega = \mathbb{R} \times \mathbb{R}$ and consider

$$y' = y^2, \qquad y(0) = 1.$$

Then, by separation of variables, we obtain the solution $y(t) = \frac{1}{1-t}$ on the interval $]-\infty < t < 1$. Hence, the solution blows up at time $T = 1$. Let us apply the MATLAB function `ode23`, which uses an adaptive time stepping strategy, to this IVP:

MATLAB

```
ode23(@blowup,[0,2],1)

function f = blowup(t,y)
f = y.^2;
```

Figure 1.1 shows that the time of blow up is detected quite accurately, and the method actually refuses to continue beyond $T = 1$, delivering the following error message:

```
Warning: Failure at t=1.001616e+00.  Unable to meet integration
tolerances without reducing the step size below the smallest
value allowed (3.552714e-15) at time t.
```

In contrast, the Euler method fails to accurately detect the time of blow up. ◇

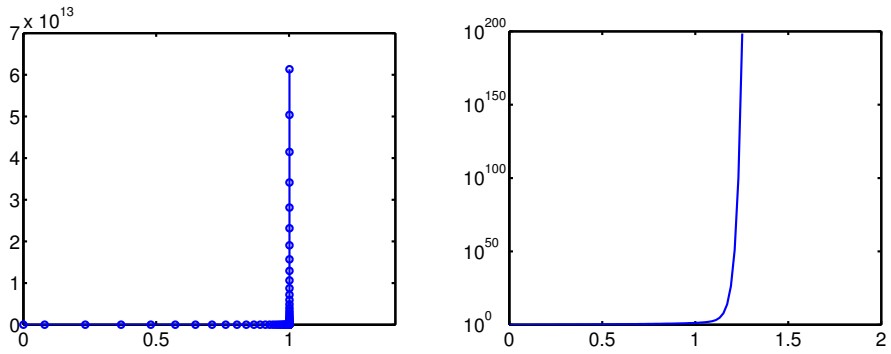


Figure 1.1. *Left plot:* `ode23` *applied to Example 1.4. Right plot: Euler method with step size* $h = 0.02$ *applied to Example 1.4.*

Collapse happens when the solution approaches the boundary of the augmented phase space Ω in finite time. It is important to note that this notion makes most sense when Ω has been maximally chosen, according to the properties of f.

Example 1.5 Let $\Omega = \mathbb{R}\times]-\infty, 0[$ and consider

$$y' = -y^{-1/2}, \qquad y(0) = 1.$$

Up to $t < 2/3$ the unique solution is given by $y(t) = (1 - 3t/2)^{2/3}$. At $t = 2/3$, however, the trajectory runs into the boundary of the phase space and collapses due to the singularity of f at 0. ◇

In the previous example, the collapse could be easily detected as the solution approaches $-\infty$ as $t \to 2/3$. A more subtle situation occurs when the solution remains bounded even when approaching the boundary of the phase space.

Example 1.6 Let $\Omega = \mathbb{R}\times]-\infty, 0[$ and consider

$$y' = \sin(1/y) - 2, \qquad y(0) = 1.$$

It can be shown that the solution remains bounded. The function $\sin(1/y)$ is continuously differentiable on the phase space, but cannot be extended continuously to a larger domain. It turns out that the solution collapses at time $T \approx 0.76741$. Figure 1.2 shows that both `ode23` and the Euler method fail to notice the collapse.

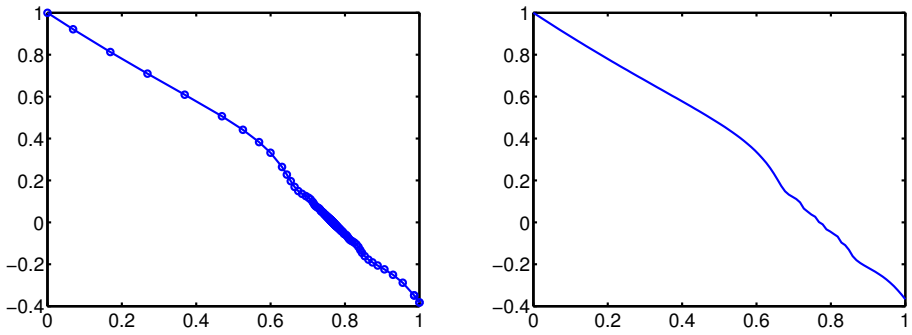


Figure 1.2. *Left plot:* `ode23` *applied to Example 1.6. Right plot: Euler method with step size* $h = 0.01$ *applied to Example 1.6.*

(At least, the very small steps taken by `ode23` when approaching T indicate a problematic behavior.) In fact, the highly oscillatory behavior causing the collapse of the solution is "averaged out" by the numerical methods. This is a good example where mathematical analysis cannot be replaced by numerics. ◇

1.2 Simple one-step methods

The **(forward) Euler method** *[méthode d'Euler (explicite/progressive)]* is the simplest numerical method for solving an IVP. Given a small step size h, the idea is to replace $\dot{\mathbf{y}}(t)$ in the differential equation $\dot{\mathbf{y}}(t) = f(t, \mathbf{y}(t))$ by the forward difference. At $t = t_0$, this yields

$$\frac{1}{h}(\mathbf{y}_1 - \mathbf{y}_0) = f(t_0, \mathbf{y}_0),$$

where $t_1 = t_0 + h$ and $\mathbf{y}_1 \approx \mathbf{y}(t_1)$. Rearranging this equation gives

$$\mathbf{y}_1 = \mathbf{y}_0 + hf(t_0, \mathbf{y}_0). \tag{1.2}$$

Repeating this procedure at $t_1, t_2, \ldots$ gives the following algorithm:

1. Choose step size $h = (T - t_0)/N$.
2. for $j = 0, 1, \ldots, N-1$ do

$$t_{j+1} = t_j + h, \qquad \mathbf{y}_{j+1} = \mathbf{y}_j + hf(t_j, \mathbf{y}_j). \tag{1.3}$$

For one-step methods, it is sufficient to specify the first step (1.2); the iteration (1.3) immediately follows.

The **backward Euler method** *[méthode d'Euler (implicite/retrograde)]* is obtained in a similar way as the forward Euler method, but using the backward difference instead of the forward difference. At $t = t_1$, this yields

$$\frac{1}{h}\big(\mathbf{y}_1 - \mathbf{y}_0\big) = f(t_1, \mathbf{y}_1),$$

where $t_1 = t_0 + h$ and $\mathbf{y}_1 \approx \mathbf{y}(t_1)$. Rearranging this equation gives

$$\mathbf{y}_1 = \mathbf{y}_0 + hf(t_1, \mathbf{y}_1).$$

In contrast to forward Euler, we need to solve a system of nonlinear equations to determine $\mathbf{y}_1$! The solution of such nonlinear systems will be discussed in some detail for more general implicit methods in Chapter 3.

As a final for a simple one-step method, we mention the **explicit trapezoidal method**

$$\mathbf{y}_1 = \mathbf{y}_0 + \frac{h}{2}\big(f(t_0, \mathbf{y}_0) + f(t_1, \mathbf{y}_0 + hf(t_0, \mathbf{y}_0))\big). \tag{1.4}$$

1.3 Consistency of one-step methods

To discuss the **local error** committed by a single step of the methods introduced above, it suffices to consider the first step:

$$\mathbf{e}(h) = \mathbf{y}(t_0 + h) - \mathbf{y}_1. \tag{1.5}$$

(Note that $\mathbf{e}(h)$ corresponds to the quantity ε_1 in *Analyse Numérique*.) As a minimal requirement, $\mathbf{e}(h)$ should converge faster to zero than h, that is

$$\mathbf{e}(h) = o(h).$$

A method satisfying this property is called **consistent**. More generally, we have the following definition.

Definition 1.7 *A method has (consistency)* **order** p *if* $\|\mathbf{e}(h)\| = O(h^{p+1})$ *holds for all sufficiently smooth* f.

It is crucial to have the exponent $p+1$ (and not p) in the definition of consistency. The underlying rationale is that we lose one power in the eventual convergence result because we have to perform $O(1/h)$ steps to reach the endpoint of an interval of constant length.

In principle, it is a simple matter to determine the order of a method by comparing the Taylor expansions of $\mathbf{y}(t_0 + h)$ and $\mathbf{y}_1$ as functions of h. Assuming that f is sufficiently smooth, we obtain for the forward Euler method that

$$\begin{aligned}\mathbf{y}(t_0 + h) &= \mathbf{y}(t_0) + h\mathbf{y}'(t_0) + O(h^2) = \mathbf{y}_0 + h\mathbf{f}(t_0, y(t_0)) + O(h^2),\\ \mathbf{y}_1 &= \mathbf{y}_0 + h\mathbf{f}(t_0, y(t_0)).\end{aligned}$$

Hence, $\|\mathbf{e}(h)\| = O(h^2)$ and therefore the forward Euler method has order 1.

For the backward Euler method, $\mathbf{y}_1(h) = \mathbf{y}_0 + hf(t_0 + h, \mathbf{y}_1(h))$. By applying the implicit function theorem, we obtain $\mathbf{y}_1'(0) = f(t_0, \mathbf{y}_1(0)) = f(t_0, \mathbf{y}_0)$. Hence, we obtain the Taylor expansion

$$\mathbf{y}_1 = \mathbf{y}_0 + h\mathbf{f}(t_0, y(t_0)) + O(h^2),$$

which implies that the backward Euler method has order 1 as well.

For (reasonable) one-step methods, consistency of order p implies convergence of order p. This has already been discussed in *Analyse Numérique*; we will have a somewhat more elaborate discussion in Section 2.2.2 on this topic.

Chapter 2

Explicit Runge-Kutta methods

The goal of this chapter is to construct explicit one-step methods of higher order. By far, the most popular methods of this type are Runge-Kutta methods, which – as we will see – are behind the MATLAB commands `ode23` and `ode45`.

2.1 A first Runge-Kutta method

The starting point for the development of simple Runge-Kutta methods is the reformulation of the IVP (1.1) on the interval $[t_0, t_0 + h]$ as the integral equation

$$\mathbf{y}(t_0 + h) = \mathbf{y}_0 + \int_{t_0}^{t_0+h} f(t, \mathbf{y}(t)) \, \mathrm{d}t. \tag{2.1}$$

When approximating the integral by a rectangle with area $h \times f(t_0, \mathbf{y}(t_0))$, we obtain

$$\mathbf{y}(t_0 + h) \approx \mathbf{y}_1 = \mathbf{y}_0 + hf(t_0, \mathbf{y}(t_0)),$$

which is the forward Euler method, so nothing new is gained. When using the midpoint rule, we approximate the integral in (2.1) by a rectangle with area $h \times f(t_0 + h/2, \mathbf{y}(t_0 + h/2))$, leading to

$$\mathbf{y}(t_0 + h) \approx \mathbf{y}_0 + hf\left(t_0 + \frac{h}{2}, \mathbf{y}(t_0 + \frac{h}{2})\right). \tag{2.2}$$

This does not look very promising either, as we do not know the value for $\mathbf{y}(t_0 + \frac{h}{2})$. In the absence of a better idea, we approximate it by one step of the forward Euler method with step size $h/2$. Inserted into (2.2), this leads to **Runge's method**:

$$\begin{aligned} \mathbf{k}_1 &= f(t_0, \mathbf{y}_0) \\ \mathbf{k}_2 &= f\left(t_0 + \frac{h}{2}, \mathbf{y}_0 + \frac{h}{2}\mathbf{k}_1\right) \\ \mathbf{y}_1 &= \mathbf{y}_0 + h\mathbf{k}_2. \end{aligned} \tag{2.3}$$

On first sight, it appears contradictory to invoke the forward Euler method for designing a higher-order method. However, the crucial point is that $\mathbf{k}_2$ gets multiplied by h, which weakens the impact of the error made when approximating $\mathbf{y}(t_0 + \frac{h}{2})$ within $\mathbf{k}_2$.

As a warm-up to the error analysis of general Runge-Kutta methods, let us have a look at the Taylor expansion of $\mathbf{y}_1$ in (2.3) as a function of h. When attempting to do so, we have to face the unpleasant situation that we have to expand both arguments of the function f appearing in $\mathbf{k}_2$. We will therefore need to make use of the **multivariate Taylor expansion**.

Remark 2.1 For a function $f \in C^k(\Omega, \mathbb{R}^d)$ with an open set $\Omega \subset \mathbb{R}^m$, the k**th derivative** $f^{(k)}(\mathbf{x})$ at $\mathbf{x} \in \Omega$ applied to $\mathbf{h}_1, \dots, \mathbf{h}_k \in \mathbb{R}^m$ is defined as

$$f^{(k)}(\mathbf{x}) \cdot (\mathbf{h}_1, \dots, \mathbf{h}_k) = \sum_{i_1, \dots, i_k = 1}^{m} \frac{\partial^k f(\mathbf{x})}{\partial x_{i_1} \cdots \partial x_{i_k}} h_{1,i_1} \dots h_{k,i_k}. \tag{2.4}$$

Thus,

$$f^{(k)}(\mathbf{x}) : \underbrace{\mathbb{R}^m \times \cdots \times \mathbb{R}^m}_{k \text{ times}} \to \mathbb{R}^d,$$

is a (symmetric) k-linear map.

For $\mathbf{h}_1 = \cdots = \mathbf{h}_k \equiv \mathbf{h}$, the sum in (2.4) can be represented in more compact form. Let $\alpha = (\alpha_1, \dots, \alpha_m)$, where each $0 \le \alpha_j \le k$ counts the differentiations with respect to the jth variable x_j. By the usual multi-index notation, we define

$$|\alpha| := \alpha_1 + \cdots + \alpha_m, \quad \frac{\partial^{|\alpha|} f}{\partial \mathbf{x}^\alpha} := \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_m^{\alpha_m}}, \quad \mathbf{h}^\alpha := h_1^{\alpha_1} \dots h_m^{\alpha_m}.$$

Each α corresponds to $\frac{k!}{\alpha!} = \frac{k!}{\alpha_1! \cdots \alpha_m!}$ terms in the sum of (2.4), and hence (2.4) is equivalent to

$$f^{(k)}(\mathbf{x}) \cdot (\mathbf{h}, \dots, \mathbf{h}) = \sum_{|\alpha| = k} \frac{k!}{\alpha!} \frac{\partial^{|\alpha|} f(\mathbf{x})}{\partial \mathbf{x}^\alpha} \mathbf{h}^\alpha. \tag{2.5}$$

◇

Theorem 2.2 *Let $f \in C^{p+1}(\Omega, \mathbb{R}^d)$ with an open set Ω and $\mathbf{x} \in \Omega$. Then*

$$f(\mathbf{x} + \mathbf{h}) = \sum_{k=0}^{p} \frac{1}{k!} f^{(k)}(\mathbf{x}) \cdot (\mathbf{h}, \dots, \mathbf{h}) + O(\|\mathbf{h}\|^{p+1})$$

for all sufficiently small $\mathbf{h} \in \mathbb{R}^m$.

Theorem 2.2 is proven by applying the usual univariate Taylor expansion in each coordinate direction separately and collecting all terms.

Example 2.3 For $d = 1$ and $p = 2$, Theorem 2.2 results in

$$\begin{aligned} f(\mathbf{x}+\mathbf{h}) &= f(\mathbf{x}) + f'(\mathbf{x})\cdot\mathbf{h} + \frac{1}{2}f''(\mathbf{x})\cdot(\mathbf{h},\mathbf{h}) + O(\|\mathbf{h}\|^3) \\ &= f(\mathbf{x}) + \nabla f(\mathbf{x})\cdot\mathbf{h} + \frac{1}{2}\mathbf{h}^\mathsf{T} H(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^3), \end{aligned}$$

where ∇f is the gradient and $H = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{i,j=1}^{m}$ is the Hessian *[matrice hessienne]* of f. ◇

For simplicity, we consider the initial value problem (1.1) for an *autonomous* differential equation:

$$\begin{cases} \mathbf{y}'(t) = f\big(\mathbf{y}(t)\big) & t \geq t_0, \\ \mathbf{y}(t_0) = \mathbf{y}_0. \end{cases} \tag{2.6}$$

Let us now recursively construct a third-order Taylor expansion for the exact solution $\mathbf{y}(t_0 + h)$. The first-order Taylor expansion is given by

$$\mathbf{y}(t_0+h) = \mathbf{y}_0 + h\mathbf{y}'(t_0) + O(h^2) = \mathbf{y}_0 + hf(\mathbf{y}_0) + O(h^2).$$

Inserting this into the differential equation (2.6) gives

$$\begin{aligned} \frac{\partial}{\partial h}\mathbf{y}(t_0+h) &= f\big(\mathbf{y}(t_0+h)\big) = f\big(\mathbf{y}_0 + hf(\mathbf{y}_0) + O(h^2)\big) \\ &= f + hf'f + O(h^2), \end{aligned}$$

where we have used the multivariate Taylor expansion from Theorem 2.2 and omitted the argument $\mathbf{y}_0$ to save space. Integration of this relation with respect to h gives the second-order Taylor expansion

$$\mathbf{y}(t_0+h) = \mathbf{y}_0 + hf + \frac{h^2}{2}f'f + O(h^3).$$

Let us repeat this process:

$$\begin{aligned} \frac{\partial}{\partial h}\mathbf{y}(t_0+h) &= f\left(\mathbf{y}_0 + hf + \frac{h^2}{2}f'f + O(h^3)\right) \\ &= f + f'\left(hf + \frac{h^2}{2}f'f\right) + \frac{1}{2}f''(hf, hf) + O(h^3) \\ &= f + hf'f + \frac{h^2}{2}\big(f'f'f + f''(f,f)\big) + O(h^3). \end{aligned}$$

By integration, the third-order Taylor expansion follows:

$$\mathbf{y}(t_0+h) = \mathbf{y}_0 + hf + \frac{h^2}{2}f'f + \frac{h^3}{6}\big(f'f'f + f''(f,f)\big) + O(h^4). \tag{2.7}$$

This will be compared with the Taylor expansion of $\mathbf{y}_1$, produced by applying (2.3) to (2.6), as a function of h:

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{y}_0 + hf\left(\mathbf{y}_0 + \frac{h}{2}f(\mathbf{y}_0)\right) \\ &= \mathbf{y}_0 + hf + \frac{h^2}{2}f'f + \frac{h^3}{8}f''(f,f) + O(h^4). \end{aligned} \tag{2.8}$$

Subtracting this from the expansion (2.7) of the exact solution gives

$$\mathbf{e}(h) = \mathbf{y}(t_0 + h) - \mathbf{y}_1 = \frac{h^3}{6}\big(f'f'f + f''(f,f)\big) - \frac{h^3}{8}f''(f,f) + O(h^4)$$

Hence, the local error satisfies $\|\mathbf{e}(h)\| = O(h^3)$, provided that all second partial derivatives of f remain bounded. This shows that the order of Runge's method is 2.

2.2 General form of explicit Runge-Kutta methods

Definition 2.4 *A method of the form*

$$\begin{aligned}
\mathbf{k}_1 &= f(t_0 + c_1 h, \mathbf{y}_0)\\
\mathbf{k}_2 &= f(t_0 + c_2 h, \mathbf{y}_0 + h a_{21}\mathbf{k}_1)\\
\mathbf{k}_3 &= f(t_0 + c_3 h, \mathbf{y}_0 + h a_{31}\mathbf{k}_1 + h a_{32}\mathbf{k}_2)\\
&\vdots\\
\mathbf{k}_s &= f\Big(t_0 + c_s h, \mathbf{y}_0 + h\sum_{\ell=1}^{s-1} a_{s\ell}\mathbf{k}_\ell\Big),\\
\mathbf{y}_1 &= \mathbf{y}_0 + h\sum_{i=1}^{s} b_i \mathbf{k}_i,
\end{aligned}$$

with all coefficients $a_{i\ell}, b_i, c_i$ being real numbers, is called an s-stage **explicit Runge-Kutta method**.

The coefficients defining a Runge-Kutta method can be organized compactly in a so-called **Butcher tableau**:

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|cccc} c_1 & 0 & \cdots & \cdots & 0 \\ c_2 & a_{21} & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array}$$

Here, A is strictly lower triangular $s \times s$ matrix and $\mathbf{b}, \mathbf{c}$ are vectors of length s. All explicit one-step methods we have discussed so far are special cases of the Runge-Kutta method:

Forward Euler: $\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$ Explicit Trapezoidal: $\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline 0 & \frac{1}{2} & \frac{1}{2} \end{array}$

Runge's method:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline 0 & 0 & 1 \end{array}$$

Another example for an explicit Runge-Kutta method is the **classical Runge-Kutta method** (RK4) defined by the tableau

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

According to Definition 2.4, this leads to the following algorithm.

Classical Runge-Kutta method:

1. Choose step size $h = (T - t_0)/N$.

2. for $j = 0, 1, \ldots, N-1$ do
   $\mathbf{k}_1 = f(t_j, \mathbf{y}_j)$
   $\mathbf{k}_2 = f(t_j + h/2, \mathbf{y}_j + h/2\,\mathbf{k}_1)$
   $\mathbf{k}_3 = f(t_j + h/2, \mathbf{y}_j + h/2\,\mathbf{k}_2)$
   $\mathbf{k}_4 = f(t_j + h, \mathbf{y}_j + h\,\mathbf{k}_3)$
   $\mathbf{y}_{j+1} = \mathbf{y}_j + \frac{h}{6}\mathbf{k}_1 + \frac{2h}{6}\mathbf{k}_2 + \frac{2h}{6}\mathbf{k}_3 + \frac{h}{6}\mathbf{k}_4$
   $t_{j+1} = t_j + h$
   end for

What is so special about the classical Runge-Kutta method? When designing an s-stage Runge-Kutta method, one goal could be to attain the maximal possible order. As we will discuss later, there is a certain limit which order can be achieved, but certainly it cannot become larger than s (see Lemma 2.5 below). One could now proceed as in Section 2.1 and compare the Taylor expansion of the exact solution with the Taylor expansion of $\mathbf{y}_1$ produced by (2.4). This appears to be a tedious task for larger values of s, but we will discuss a clever way of organizing it in the the following section. Matching both expansions up to a certain order leads to a system of nonlinear equations in the coefficients defining the Runge-Kutta method. Finding all solutions to this system by hand is a nearly impossible task, except for tiny s. For example, the case $s = 4$ is covered in Section II.1 of [HNW], giving a fairly general parametrization of 4-stage explicit Runge-Kutta methods having order 4.

2.2.1 Order conditions for Runge-Kutta methods

The ultimate goal of this section is to develop an elegant way that avoids the tediousness of Taylor expansions when analyzing the order of a Runge-Kutta method. Before, we will make some basic observations.

Lemma 2.5 *If an s-stage explicit Runge-Kutta method has order p for all $f \in C^\infty(\Omega, \mathbb{R}^d)$, then $p \leq s$.*

Proof. We apply the Runge-Kutta method to the scalar IVP $y'(t) = y(t), y(0) = y_0 = 1$. Then each stage k_i is a polynomial of degree at most $i-1$ in h. Hence, y_1 is a polynomial of degree at most s in h. Consequently, the $(s+1)$th term $\frac{h^{s+1}}{(s+1)!}$ in the Taylor expansion of the exact solution $y(h) = e^h$ cannot be matched by y_1. Hence, the order of the Runge-Kutta method is at most s. ∎

Lemma 2.6 *An explicit Runge-Kutta method is consistent for all $f \in C(\Omega, \mathbb{R}^d)$ if and only if*

$$\sum_{i=1}^{s} b_i = 1. \tag{2.9}$$

Proof. The first term in the expansion of $\mathbf{y}_1$ takes the form

$$\mathbf{y}_1 = \mathbf{y}_0 + h \sum_{i=1}^{s} b_i f(t_0, \mathbf{y}_0) + \cdots$$

This matches the first two terms in the Taylor expansion (2.7) if and only if (2.9) is satisfied. ∎

The analysis greatly simplifies if we only need to consider autonomous IVPs, as we did in Section 2.1. There is a simple trick to transform (1.1) to autonomous form by appending t to the variables:

$$\begin{pmatrix} t \\ \mathbf{y} \end{pmatrix}' = \begin{pmatrix} 1 \\ f(t, \mathbf{y}) \end{pmatrix}, \qquad \begin{pmatrix} t \\ \mathbf{y} \end{pmatrix}_{t=t_0} = \begin{pmatrix} t_0 \\ \mathbf{y}_0 \end{pmatrix}. \tag{2.10}$$

This trick makes perfect sense if we have a correspondence between the Runge-Kutta method applied to (2.10) and applied to (1.1). Ideally, the approximation produced by one step of the Runge-Kutta method applied to (2.10) takes the form

$$\begin{pmatrix} t_0 + h \\ \mathbf{y}_1 \end{pmatrix}.$$

A Runge-Kutta method having this property is called **invariant under autonomization**.

Lemma 2.7 *An explicit Runge-Kutta method is invariant under autonomization if and only if it is consistent and satisfies*

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \qquad i = 1, \ldots, s.$$

Proof. The stages of the Runge-Kutta method applied to (2.10) take the form

$$\begin{aligned}\begin{pmatrix}\theta_i\\ \hat{\mathbf{k}}_i\end{pmatrix} &= \begin{pmatrix}1\\ f\Big(t_0 + h\sum_j a_{ij}\theta_j, \mathbf{y}_0 + h\sum_j a_{ij}\hat{\mathbf{k}}_j\Big)\end{pmatrix}\\ &= \begin{pmatrix}1\\ f\Big(t_0 + h\sum_j a_{ij}, \mathbf{y}_0 + h\sum_j a_{ij}\hat{\mathbf{k}}_j\Big).\end{pmatrix}\end{aligned}$$

Clearly, we have $\hat{\mathbf{k}}_i = \hat{k}_i$ for general f if and only if $\sum_j a_{ij} = c_i$. The approximate solution is then given by

$$\begin{pmatrix}t_0 + h\sum_i b_i\theta_i\\ \mathbf{y}_0 + h\sum_i b_i\mathbf{k}_i\end{pmatrix} = \begin{pmatrix}t_0 + h\sum_i b_i\\ \mathbf{y}_1\end{pmatrix}.$$

Hence, invariance holds if we additionally require that $\sum_i b_i = 1$, which – by Lemma 2.6 – is equivalent to the consistency of the Runge-Kutta method. □

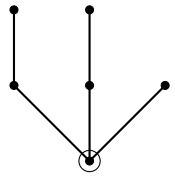
Letting $e = (1, \dots, 1)^{\mathsf{T}} \in \mathbb{R}^s$ denote the vector of all ones, the conditions of Lemma 2.6 and Lemma 2.7 can be compactly written as

$$\mathbf{b}^{\mathsf{T}}\mathbf{e} = 1, \qquad \mathbf{c} = A\mathbf{e}.$$

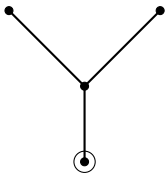
For the rest of this section, we will assume that these conditions are satisfied and only consider autonomous IVPs.

Elementary differentials and rooted trees. The biggest problem when attempting to verify (high) orders of Runge-Kutta methods is to keep track of the terms appearing in the Taylor expansions of the exact and the approximate solutions. As can be seen from (2.7) and (2.8), these terms are elementary differentials of f. Counting the number of the various elementary differentials appearing in the expansion is a combinatorial problem, for which we will use the concept of rooted trees.

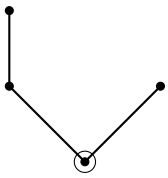
Roughly speaking, an elementary differential is a multilinear form that is fully described by the recursive structure of its arguments, a more precise definition will be given below. To give a specific example, let us consider the differential $f'''(f'f, f'f, f)$. We can deduce the presence of the third derivative from the fact that three arguments need to be provided. To detail the specific structure of the arguments, we can use rooted trees, such as:



for $f'''(f'f, f'f, f)$



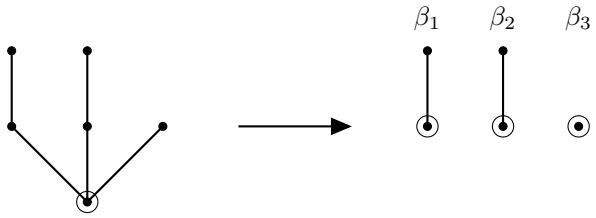
for $f'f''(f, f)$



for $f''(f'f, f)$

Each node of the rooted tree corresponds to a derivative of f at $\mathbf{y}_0$, with the order k of the derivative determined by the number of children. The substitution of the arguments proceeds recursively starting at the root $\odot$. Note that the order of the children is *not* important, due the symmetry of the multilinear form.

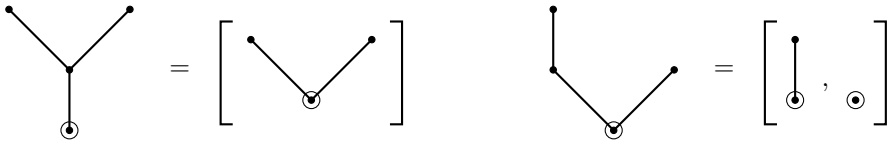
When the root $\odot$ is deleted from a tree β then β decomposes into a forest of k trees $\beta_1, \ldots, \beta_k$, whose roots correspond to the k children of $\odot$. For example:



In the other direction, the tree β can be composed from an unordered tuple of subtrees:

$$\beta = [\beta_1, \ldots, \beta_k], \qquad \#\beta = 1 + \#\beta_1 + \cdots + \#\beta_k,$$

where $\#\beta$ denotes the number of the nodes in a tree β. For creating a tree consisting solely of the root, we use the convention $\odot = []$. Two examples for the composition of trees:



This decomposition can be continued recursively until the subtrees consist of roots only. Two examples:

$$\begin{array}{ll} f'f''(f, f) & \text{corresponds to} \quad [[\odot, \odot]], \\ f''(f'f, f) & \text{corresponds to} \quad [[\odot], \odot]. \end{array}$$

With the notation defined above, we are ready to define elementary differentials.

Definition 2.8 *The* **elementary differential** $f^{(\beta)}(\mathbf{y}_0)$ *corresponding to a rooted tree* $\beta = [\beta_1, \dots, \beta_k]$ *is defined recursively as*

$$f^{(\beta)}(\mathbf{y}_0) = f^{(k)} \cdot \Big(f^{(\beta_1)}(\mathbf{y}_0), \dots, f^{(\beta_k)}(\mathbf{y}_0) \Big).$$

In the definition above, the argument $\mathbf{y}_0$ is given for clarity; we will (again) often omit it in our subsequent considerations.

Remark 2.9 The Hopf algebra of rooted trees does not only play a role in the theory of Runge–Kutta methods, but also in noncommutative geometry and renormalization methods in quantum field theory. Quite remarkably, it was developed first in the context of numerical analysis, by John Butcher. We refer to [Brouder, Ch. Trees, renormalization and differential equations. BIT 44 (2004), no. 3, 425–438] for a very accessible exposition of these connections. ◇

Taylor expansion of the exact solution. Let us recall the Taylor expansion (2.7):

$$\mathbf{y}(t_0 + h) = \mathbf{y}_0 + hf + \frac{h^2}{2} f'f + \frac{h^3}{6} \big(f'f'f + f''(f,f) \big) + O(h^4).$$

A closer inspection suggests that the terms for h^k involve all elementary differentials corresponding to rooted trees with k nodes. Lemma 2.10 below confirms this impression in the general case. For this purpose, we need to introduce two combinatorial quantities. The **factorial of a tree** $\beta = [\beta_1, \dots, \beta_k]$ is recursively defined by

$$\beta! = (\#\beta)\beta_1!\beta_2! \cdots \beta_k!.$$

Moreover, we define recursively

$$\alpha_\beta = \frac{\delta_\beta}{k!} \alpha_{\beta_1} \alpha_{\beta_2} \cdots \alpha_{\beta_k},$$

where δ_β denotes the number of different possibilities for associating an an *ordered* k-tuple with the unordered k-tuple $\beta = [\beta_1, \dots, \beta_k]$.

Lemma 2.10 *Let* $f \in C^p(\Omega, \mathbb{R}^d)$ *with an open set* Ω *and* $\mathbf{y}_0 \in \Omega$. *Then*

$$\mathbf{y}(t_0 + h) = \mathbf{y}_0 + \sum_{\#\beta \le p} \frac{h^{\#\beta}}{\beta!} \alpha_\beta f^{(\beta)}(\mathbf{y}_0) + O(h^{p+1}).$$

Proof. By induction over p. For $p = 0$, the statement trivially holds. We now show the expansion for $p + 1$, assuming that it holds for p. The proof proceeds, as in the analysis of Section 2.1, by inserting the pth order expansion into the differential equation and using the multivariate Taylor expansion. Setting $\mathbf{h} =$

$\sum_{\#\beta\le p} \frac{h^{\#\beta}}{\beta!}\alpha_\beta f^{(\beta)}$, we obtain

$$\frac{\partial}{\partial h}\mathbf{y}(t_0{+}h) = f(\mathbf{y}_0{+}th) = f\left(\mathbf{y}_0+\mathbf{h}+O(h^{p+1})\right) = \sum_{k=0}^{p}\frac{1}{k!}f^{(k)}(\mathbf{h},\dots,\mathbf{h}){+}O(h^{p+1}).$$

Using the multilinearity of $f^{(k)}$ and omitting terms smaller than h^p, we obtain

$$\begin{aligned}
&\frac{1}{k!}f^{(k)}(\mathbf{h},\dots,\mathbf{h})\\
&= \frac{1}{k!}\sum_{\#\beta_1\le p}\cdots\sum_{\#\beta_k\le p}\frac{h^{\#\beta_1+\cdots+\#\beta_k}}{\beta_1!\cdots\beta_k!}\alpha_{\beta_1}\cdots\alpha_{\beta_k}f^{(k)}\big(f^{(\beta_1)},\dots,f^{(\beta_k)}\big)\\
&= \frac{1}{k!}\sum_{\#\beta_1+\cdots+\#\beta_k\le p}\frac{h^{\#\beta_1+\cdots+\#\beta_k}}{\beta_1!\cdots\beta_k!}\alpha_{\beta_1}\cdots\alpha_{\beta_k}f^{(k)}\big(f^{(\beta_1)},\dots,f^{(\beta_k)}\big)+O(h^{p+1})\\
&= \sum_{\substack{\#\beta\le p+1\\ \beta=[\beta_1,\dots,\beta_k]}}\frac{\#\beta\cdot h^{\#\beta-1}}{\beta!}\underbrace{\frac{\delta_\beta}{k!}\alpha_{\beta_1}\cdots\alpha_{\beta_k}}_{=\alpha_\beta}f^{(\beta)}+O(h^{p+1}),
\end{aligned}$$

where we have used in the last step that the symmetry of $f^{(k)}$ allows to use unordered tuples $[\beta_1,\dots,\beta_k]$ instead of ordered tuples. This introduces the extra factor δ_β, which counts the number of ordered tuples associated with $[\beta_1,\dots,\beta_k]$. To sum up, we obtain

$$\frac{\partial}{\partial h}\mathbf{y}(t_0+h) = \sum_{\#\beta\le p+1}\frac{\#\beta\cdot h^{\#\beta-1}}{\beta!}\alpha_\beta f^{(\beta)}+O(h^{p+1}).$$

Integrating this relation gives

$$\mathbf{y}(t_0+h) = \mathbf{y}_0+\sum_{\#\beta\le p+1}\frac{h^{\#\beta}}{\beta!}\alpha_\beta f^{(\beta)}+O(h^{p+2}).$$

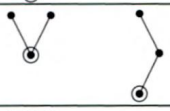
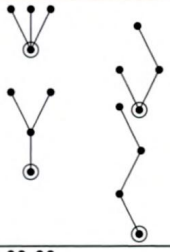
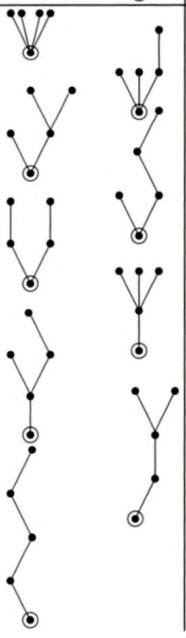
This corresponds exactly to the claimed expansion for $p+1$. □

Examples for the coefficients $\beta!$ and α_β can be found in Table 2.1.

Taylor expansion of the numerical solution. The expansion of the numerical solution $\mathbf{y}_1$ produced by the Runge-Kutta method can also be represented with the help of rooted trees. For a rooted tree $\beta=[\beta_1,\dots,\beta_k]$, we recursively define the vector $A^{(\beta)}\in\mathbb{R}^s$ via its components:

$$A_i^{(\beta)} = \left(A\cdot A^{(\beta_1)}\right)_i\cdots\left(A\cdot A^{(\beta_k)}\right)_i,\qquad i=1,\dots,s.$$

Table 2.1. *Rooted trees up to order* 5. *Table taken from Page 150 of [DB].*

$\#\beta$	β	rooted tree	$\beta!$	α_β	$f^{(\beta)}$	$\mathcal{A}_i^{(\beta)}$
1	β_{11}		1	1	f	1
2	β_{21}		2	1	$f'f$	c_i
3	β_{31}		3	$\frac{1}{2}$	$f''(f,f)$	c_i^2
	β_{32}		6	1	$f'f'f$	$\sum_j a_{ij}c_j$
4	β_{41}		4	$\frac{1}{6}$	$f'''(f,f,f)$	c_i^3
	β_{42}		8	1	$f''(f'f,f)$	$\sum_j c_i a_{ij}c_j$
	β_{43}		12	$\frac{1}{2}$	$f'f''(f,f)$	$\sum_j a_{ij}c_j^2$
	β_{44}		24	1	$f'f'f'f$	$\sum_{jk} a_{ij}a_{jk}c_k$
5	β_{51}		5	$\frac{1}{24}$	$f^{IV}(f,f,f,f)$	c_i^4
	β_{52}		10	$\frac{1}{2}$	$f'''(f'f,f,f)$	$\sum_j c_i^2 a_{ij}c_j$
	β_{53}		15	$\frac{1}{2}$	$f''(f''(f,f),f)$	$\sum_j c_i a_{ij}c_j^2$
	β_{54}		30	1	$f''(f'f'f,f)$	$\sum_{jk} c_i a_{ij}a_{jk}c_k$
	β_{55}		20	$\frac{1}{2}$	$f''(f'f,f'f)$	$\left(\sum_j a_{ij}c_j\right)^2$
	β_{56}		20	$\frac{1}{6}$	$f'f'''(f,f,f)$	$\sum_j a_{ij}c_j^3$
	β_{57}		40	1	$f'f''(f'f,f)$	$\sum_{jk} a_{ij}c_j a_{jk}c_k$
	β_{58}		60	$\frac{1}{2}$	$f'f'f''(f,f)$	$\sum_{jk} a_{ij}a_{jk}c_k^2$
	β_{59}		120	1	$f'f'f'f'f$	$\sum_{jkl} a_{ij}a_{jk}a_{kl}c_l$

Again, this definition is independent of the order of the children $\beta_1, \ldots, \beta_k$. Note that

$$A^{(\odot)} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^s.$$

Further examples for $A^{(\beta)}$ can be found in Table 2.1, where the assumed relation $\mathbf{c} = A\mathbf{e}$ has been used extensively to simplify the formulas.

Lemma 2.11 *Let $f \in C^p(\Omega, \mathbb{R}^d)$ with an open set Ω and $\mathbf{y}_0 \in \Omega$. Then the approximation $\mathbf{y}_1$ obtained by applying the Runge-Kutta method (Definition 2.4) to the autononomous IVP (2.6) satisfies*

$$\mathbf{y}_1 = \mathbf{y}_0 + \sum_{\#\beta \le p} h^{\#\beta} \alpha_\beta \cdot b^{\mathsf{T}} A^{(\beta)} f^{(\beta)}(\mathbf{y}_0) + O(h^{p+1}).$$

Proof. Since

$$\mathbf{y}_1 = \mathbf{y}_0 + h \sum_{i=1}^{s} b_i \mathbf{k}_i,$$

it suffices to show the Taylor expansions

$$\mathbf{k}_i = \sum_{\#\beta \le p} h^{\#\beta - 1} \alpha_\beta A_i^{(\beta)} f^{(\beta)} + O(h^p), \qquad i = 1, \dots, s \tag{2.11}$$

for the stages $\mathbf{k}_i$. The proof of (2.11) proceeds by induction over p. It clearly holds for $p = 0$ and we now aim to derive (2.11) for $p+1$, assuming that it holds for p. We then have

$$\begin{aligned}
\mathbf{k}_i &= f\left(\mathbf{y}_0 + h \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j\right) \\
&= f\left(\mathbf{y}_0 + \sum_{j=1}^{i-1} a_{ij} \sum_{\#\beta \le p} h^{\#\beta} \alpha_\beta A_i^{(\beta)} f^{(\beta)} + O(h^{p+1})\right) \\
&= f\left(\mathbf{y}_0 + \sum_{\#\beta \le p} h^{\#\beta} \alpha_\beta \Big(A \cdot A^{(\beta)}\Big)_i f^{(\beta)} + O(h^{p+1})\right).
\end{aligned}$$

Setting $\mathbf{h} = \sum_{\#\beta \le p} h^{\#\beta} \alpha_\beta \Big(A \cdot A^{(\beta)}\Big)_i f^{(\beta)}$, the multivariate Taylor expansion combined with the multilinearity of $f^{(k)}$ yield

$$\begin{aligned}
\mathbf{k}_i &= \sum_{k=0}^{p} \frac{1}{k!} f^{(k)} \cdot (\mathbf{h}, \dots, \mathbf{h}) + O(h^{p+1}) \\
&= \sum_{k=0}^{p} \frac{1}{k!} \sum_{\#\beta_1 + \dots + \#\beta_k \le p} h^{\#\beta_1 + \dots + \#\beta_k} \cdot \alpha_{\beta_1} \cdots \alpha_{\beta_k} \Big(A \cdot A^{(\beta_1)}\Big)_i \cdots \Big(A \cdot A^{(\beta_k)}\Big)_i \\
&\qquad\qquad \cdots f^{(k)} \cdot \Big(f^{(\beta_1)}, \dots f^{(\beta_k)}\Big) + O(h^{p+1}) \\
&= \sum_{k=0}^{p} \sum_{\substack{\#\beta \le p+1 \\ \beta = [\beta_1, \dots, \beta_k]}} h^{\#\beta - 1} \cdot \frac{\delta_\beta}{k!} \alpha_{\beta_1} \cdots \alpha_{\beta_k} \cdot A_i^{(\beta)} f^{(\beta)} + O(h^{p+1}) \\
&= \sum_{\#\beta \le p+1} h^{\#\beta - 1} \alpha_\beta \cdot A_i^{(\beta)} f^{(\beta)} + O(h^{p+1}).
\end{aligned}$$

This completes the proof, as the last line corresponds exactly to (2.11) for $p+1$. □

Order conditions. After all these preparations, we can now combine Lemma 2.10 and Lemma 2.11 to obtain a pleasingly elegant necessary and sufficient condition for a Runge-Kutta method having order p.

Theorem 2.12 *A Runge-Kutta method is of order p if and only if*

$$\mathbf{b}^{\mathsf{T}} A^{(\beta)} = \frac{1}{\beta!} \tag{2.12}$$

holds for all rooted trees β of order $\#\beta \le p$.

Proof. The sufficiency of condition (2.12) follows immediately from comparing the expansions in Lemma 2.10 and Lemma 2.11. For showing that (2.12) is also necessary, one needs to show that the elementary differentials for different trees are linearly independent. This is proven, e.g., in Lemma 4.25 of [DB]. □

Example 2.13 For $p=4$, Theorem 2.12 leads to the following conditions:

$$\begin{array}{|rcl|}\hline 1 &=& \sum_{i=1}^{s} b_i \\ \hline \frac{1}{2} &=& \sum_i b_i c_i \\ \hline \frac{1}{3} &=& \sum_i b_i c_i^2 \\ \frac{1}{6} &=& \sum_{i,j} b_i a_{ij} c_j \\ \hline \end{array}, \qquad \begin{array}{|rcl|}\hline \frac{1}{4} &=& \sum_i b_i c_i^3 \\ \hline \frac{1}{8} &=& \sum_{i,j} b_i c_i a_{ij} c_j \\ \frac{1}{12} &=& \sum_{i,j} b_i a_{ij} c_j^2 \\ \frac{1}{24} &=& \sum_{i,j,k} b_i a_{ij} a_{jk} c_k \\ \hline \end{array}$$

It is a simple exercise to verify that RK4 satisfies this conditions and therefore has order 4. ◇

From Example 2.13 one may obtain the misleading impression that the number of conditions grows only slowly with p. In fact, for $p=10$ one has 1205 conditions and for $p=20$ one has 20247374 conditions!

2.2.2 Convergence results

By definition, the local error $\mathbf{e}(h) = \mathbf{y}(t_0+h) - \mathbf{y}_1$ of a Runge-Kutta method of order p satisfies $\|\mathbf{e}(h\| \le Ch^{p+1}$ for some constant C.

We are now concerned with analyzing the **global error**, which is the error of the computed solution after *several* steps: $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \ldots$. For this purpose, we will write

$$\mathbf{y}_{i+1} = \mathbf{y}_i + h_i \Phi(t_i, \mathbf{y}_i, h_i), \qquad h_i = t_{i+1} - t_i,$$

where Φ is the **increment function** of the method. Every one-step method can be written in this form. Our task is to estimate the **global error**

$$\mathbf{E} = \mathbf{y}(T) - \mathbf{y}_N, \quad \text{with} \quad T = t_N. \tag{2.13}$$

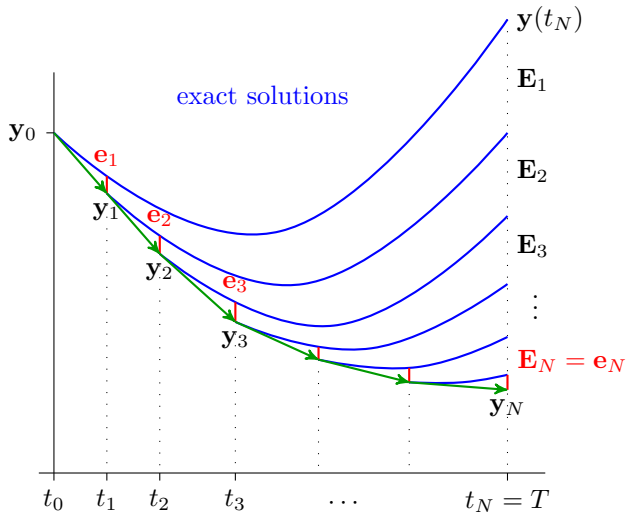


Figure 2.1. *Lady Windemere's fan [l'éventail de Lady Windermere].*

This estimate is found by transporting all local errors $\mathbf{e}_i = \mathbf{y}(t_j) - \mathbf{y}_i$, which satisfy

$$\|\mathbf{e}_i\| \le Ch_{i-1}^{p+1}, \tag{2.14}$$

to the end point t_N and adding them up. The principle is illustrated in Figure 2.1.

The following lemma can be used to estimate the magnification of the error during the transport.

Lemma 2.14 *Suppose that* $\mathbf{y}(t)$ *and* $\mathbf{v}(t)$ *are solutions to* $\mathbf{y}' = f(t, \mathbf{y})$ *with initial values* $\mathbf{y}(t_0) = \mathbf{y}_0$ *and* $\mathbf{v}(t_0) = \mathbf{v}_0$*, respectively. If*

$$\|f(t, \mathbf{v}) - f(t, \mathbf{y})\| \le L\|\mathbf{v} - \mathbf{y}\| \tag{2.15}$$

then we have the error estimate

$$\|\mathbf{v}(t) - \mathbf{y}(t)\| \le e^{L(t-t_0)} \cdot \|\mathbf{v}_0 - \mathbf{y}_0\|. \tag{2.16}$$

Proof. Integration of $\mathbf{v}' - \mathbf{y}' = f(t, \mathbf{v}) - f(t, \mathbf{y})$ with respect to t gives

$$\mathbf{v} - \mathbf{y} = \mathbf{v}_0 - \mathbf{y}_0 + \int_{t_0}^{t} \big(f(s, \mathbf{v}(s)) - f(s, \mathbf{y}(s))\big)\, \mathrm{d}s.$$

Hence,

$$\|\mathbf{v} - \mathbf{y}\| \le \|\mathbf{v}_0 - \mathbf{y}_0\| + L \int_{t_0}^{t} \|\mathbf{v}(s) - \mathbf{y}(s)\|\, \mathrm{d}s. \tag{2.17}$$

We can apply the (continuous) lemma of Gronwall to this situation (see Lemma 2.15 below), with $u(t) = \|\mathbf{v}(t) - \mathbf{y}(t)\|$, $\rho = \|\mathbf{v}_0 - \mathbf{y}_0\|$, and $w(t) \equiv L$. This immediately gives (2.16). ∎

Lemma 2.15 (Lemma of Gronwall) *Let $u, w \in C([t_0, T])$ with w nonnegative, and let $\rho \geq 0$. Then*

$$u(t) \leq \rho + \int_{t_0}^{t} u(s)w(s)\, ds \quad \text{for all } t \in [t_0, T]$$

implies the estimate

$$u(t) \leq \rho \exp\left(\int_{t_0}^{t} w(s)\, ds\right) \quad \text{for all } t \in [t_0, T].$$

Proof. EFY, cf. Ex.1,3(b). ∎

Using Lemma 2.14, it remains to add the transported local errors in order to obtain an estimate for the global error.

Theorem 2.16 *Let U be a neighborhood of $\{(t, \mathbf{y}(t)) : t_0 \leq t \leq T\}$ such that the local error estimate* (2.14) *and the estimate*

$$\left\| \frac{\partial f}{\partial y} \right\| \leq L \tag{2.18}$$

hold in U. Then the global error (2.13) *satisfies*

$$\|\mathbf{E}\| \leq h^p \frac{C}{L} \left(e^{L(T-t_0)} - 1\right), \tag{2.19}$$

where $h = \max_i h_i$ is small enough for the numerical solution to remain in U.

Proof. Let $\mathbf{E}_i$ denote the ith local error $\mathbf{e}_i$ transported to the end point, see Figure 2.1. Then Lemma 2.14 applied at time t_i with initial values $\mathbf{y}_i$ vs. $\mathbf{y}(t_i)$ yields

$$\|\mathbf{E}_i\| \leq e^{L(T-t_i)} \cdot \|\mathbf{e}_i\| \leq C e^{L(T-t_i)} h_{i-1}^{p+1}.$$

Using $h_{i-1}^{p+1} \leq h^p h_{i-1}$ yields

$$\|\mathbf{E}\| \leq \sum_{i=1}^{N} \|\mathbf{E}_i\| \leq C h^p \sum_{i=1}^{N} e^{L(T-t_i)} h_{i-1} \leq C h^p \int_{t_0}^{T} e^{L(T-s)}\, \mathrm{d}s.$$

In the last inequality, we have used that the sum is actually a lower Darboux sum *[somme de Darboux inférieure]* for the integral on the right-hand side, see Figure 2.2. The elementary fact $\int_{t_0}^{T} e^{L(T-s)}\, \mathrm{d}s = \frac{1}{L}\left(e^{L(T-t_0)} - 1\right)$ concludes the proof. ∎

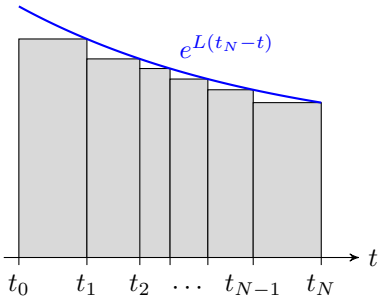


Figure 2.2. *Lower Darboux sum used in the proof of Theorem 2.16.*

Clearly, Theorem 2.16 indicates that we may run into severe problems (that is, exponentially growing error) as $T - t_0$ grows. Still, ODEs are routinely integrated over relatively long time, especially in molecular dynamics and astronomy. However, this may require the use of special methods, see [HLW].

Remark 2.17* As the Lipschitz constant L is positive, Theorem 2.16 always implies an exponential growth of the error. This is clearly not very satisfactory for differential equations such as $y' = -y$, where both the solution and the transported error are *damped* as t increases. The guilty party is Lemma 2.14, which is too pessimistic about the effect of transport on the error. We can derive a different estimate as follows. First, note that

$$\begin{aligned}\|\mathbf{v}(t+h) - \mathbf{y}(t+h)\| &= \big\|\mathbf{v}(t) - \mathbf{y}(t) + h\big(f(t,\mathbf{v}) - f(t,\mathbf{y})\big)\big\| + O(h^2)\\ &\le \left(\max_{\mathbf{x}\in V}\left\|I + h\frac{\partial f}{\partial \mathbf{y}}(t,\mathbf{x})\right\|\right)\cdot\|\mathbf{v}(t) - \mathbf{y}(t)\| + O(h^2),\end{aligned}$$

where we applied the mean value theorem and V is a connected neighborhood containing both $\mathbf{v}$ and $\mathbf{y}$. Setting $f(t) := \|\mathbf{v}(t) - \mathbf{y}(t)\|$, we therefore obtain

$$\frac{m(t+h) - m(t)}{h} \le \left(\max_{\mathbf{x}\in V}\frac{\left\|I + h\frac{\partial f}{\partial \mathbf{y}}(t,\mathbf{x})\right\| - 1}{h}\right) m(t) + O(h). \tag{2.20}$$

As $h \to 0$, the factor on the right-hand side features a quantity known as the logarithmic "norm" of a matrix.

Definition 2.18 *Let $A \in \mathbb{C}^{n\times n}$. Then*

$$\mu(A) := \lim_{\substack{h\to 0\\ h>0}} \frac{\|I + hA\| - 1}{h}$$

is called the **logarithmic norm** *[norme logarithmique] of A.*

Of course, Definition 2.18 depends on the choice of matrix norm $\|\cdot\|$. For the matrix 2-norm, it is not hard to show that $\mu(A) = \frac{1}{2}\lambda_{\max}(A + A^*)$.

From (2.20) it now follows that the upper Dini derivative m'_+ satisfies

$$m'_+(t) \leq \max_{\mathbf{x}\in V} \mu\left(\frac{\partial f}{\partial \mathbf{y}}(t,\mathbf{x})\right) m(t).$$

Integrating this inequality gives (2.17) but with $L = \max_{\mathbf{x}\in V} \mu\left(\frac{\partial f}{\partial \mathbf{y}}(t,\mathbf{x})\right)$ replacing the Lipschitz constant. As a consequence, we get an improved variant of Theorem 2.16. The assumption (2.18) is replaced by

$$\mu\left(\frac{\partial f}{\partial \mathbf{y}}\right) \leq L$$

and the global error estimate (2.19) reads as

$$\|\mathbf{E}\| \leq h^p \frac{C'}{L}\left(e^{L(T-t_0)} - 1\right).$$

Here, $C' = C$ for $L > 0$. For $L < 0$, one has to adjust the Darboux sum in the proof of Theorem 2.16 and obtains $C' = Ce^{-Lh}$. ◇

Chapter 3

Implicit Runge-Kutta methods

Although the family of explicit Runga-Kutta methods is quite rich, they may be ineffective for some (particularly hard) problems. Indeed, we will see that *no* explicit method is suitable for so called stiff problems, which frequently arise in practice, in particular from the spatial discretization of time-dependent partial differential equations. It turns out that implicit methods are much more effective for stiff problems. However, we will see that the price one has to pay for going implicit is very high; function evaluations are replaced by the solution of nonlinear systems!

Example 3.1 To solve the IVP

$$\dot{y}(t) = 500\, y^2(1-y), \quad y(0) = 1/100,$$

we make use of the MATLAB function `ode23s`, which is based on Rosenbrock methods – a variation of implicit Runge-Kutta methods discussed in Section 3.5. For this purpose, we need to define the function as well as its derivative (Jacobian) with respect to y (or an approximation of it):

```
fun = @(t,y) 500*y^2*(1-y); funjac = @(t,y) 1000*y*(1-y) - 500*y^2;
```

Additionally to the usual tolerances for the time stepping procedure, the derivative also needs to be declared in the options:

```
opt = odeset( 'reltol', 0.1, 'abstol', 0.001, 'Jacobian', funjac );
```

Finally, the integration is performed on the interval $[0, 1]$:

```
[t,x] = ode23s(fun, [0,1], 0.01, opt);
plot(t,x);
```

From Figure 3.1, it is clear that `ode23s` requires much less time steps and function evaluations compared to the explicit Runge-Kutta method in `ode23` with the same options. Moreover, `ode23s` appears to be significantly more accurate than `ode23`. ◇

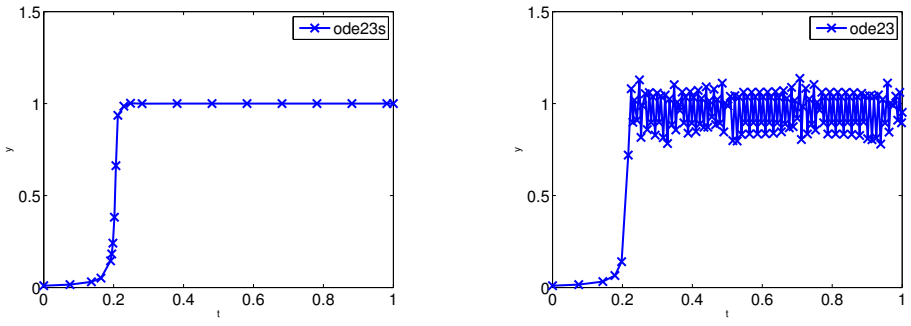


Figure 3.1. Results of `ode23s` and `ode23` applied to Example 3.1.

3.1 Stability concepts

The notion of stability is crucial to understand the limitations of explicit methods for stiff problems. We consider a linear homogeneous IVP

$$\dot{\mathbf{y}}(t) = G\mathbf{y}(t), \qquad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{3.1}$$

for a $d \times d$ matrix G. The IVP (3.1) is called

- **asymptotically stable** if $\|\mathbf{y}(t)\| \to 0$ as $t \to \infty$ for *all* initial values $\mathbf{y}_0 \in \mathbb{R}^d$;
- **stable** if there is a constant C (independent of t and $\mathbf{y}_0$) such that $\|\mathbf{y}(t)\| < C\|\mathbf{y}_0\|$ holds for all $t \geq t_0$ and $\mathbf{y}_0 \in \mathbb{R}^d$;
- **unstable**, otherwise.

Stability has a number of important consequences. For example if we consider an inhomogeneous problem with undergoing a perturbation of the initial value:

$$\begin{aligned} \dot{\mathbf{y}}_1(t) &= C\mathbf{y}_1(t), \qquad \mathbf{y}_1(t_0) = \mathbf{y}_0, \\ \dot{\mathbf{y}}_2(t) &= C\mathbf{y}_2(t), \qquad \mathbf{y}_2(t_0) = \mathbf{y}_0 + \triangle\mathbf{y}_0. \end{aligned}$$

then stability implies $\|\mathbf{y}_1(t) - \mathbf{y}_2(t)\| \leq C\|\triangle\mathbf{y}_0\|$ and asymptotic stability even implies that $\|\mathbf{y}_1(t) - \mathbf{y}_2(t)\| \to 0$. That is, the impact of a perturbation is bounded or vanishes in the long time, respectively.

Let us define the **matrix exponential**

$$e^B := \sum_{n=0}^{\infty} \frac{1}{n!} B^n, \tag{3.2}$$

which is absolutely convergent for any $B \in \mathbb{C}^{d\times d}$. Then the solution of (3.1) is given by

$$\mathbf{y}(t) = e^{G(t-t_0)}\mathbf{y}_0.$$

Hence, asymptotic stability is equivalent to $\|e^{Gt}\| \to 0$ as $t \to \infty$. Stability is equivalent to $\|e^{Gt}\| \le C$ for all $t > 0$. The following theorem gives a complete characterization of (asymptotic) stability in terms of the eigenvalues of G. Note that an eigenvalue λ of G is called **semi-simple** if its algebraic and geometric multiplicities are equal or, equivalently, if all blocks in the Jordan canonical form of G associated with λ are 1×1.

Theorem 3.2 *The IVP* (3.1) *is asymptotically stable if and only if all eigenvalues λ of G satisfy* $\mathsf{Re}(\lambda) < 0$.

The IVP (3.1) *is stable if and only if all eigenvalues λ of G satisfy* $\mathsf{Re}(\lambda) \le 0$ and *if every eigenvalue λ with* $\mathsf{Re}(\lambda) = 0$ *is semi-simple.*

Proof. For simplicity, we assume that G is diagonalizable. (The general case requires to study the Jordan canonical form of A, which is beyond the scope of this lecture.) Then there is an invertible matrix P such that

$$G = P\Lambda P^{-1} \quad \text{with} \quad \Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{pmatrix},$$

where $\lambda_1, \dots, \lambda_d$ are the eigenvalues of G. Then

$$\begin{aligned} e^{Gt} &= \sum_{n=0}^{\infty} \frac{1}{n!} G^n t^n = \sum_{n=0}^{\infty} \frac{1}{n!} (P\Lambda P^{-1})^n t^n = \sum_{n=0}^{\infty} \frac{1}{n!} P\Lambda^n P^{-1} t^n \\ &= P\, e^{\Lambda t} P^{-1} = P \begin{pmatrix} e^{\lambda_1 t} & & \\ & \ddots & \\ & & e^{\lambda_d t} \end{pmatrix} P^{-1}. \end{aligned}$$

Using that $e^{\lambda t} \to 0$ converges to zero if and only if $\mathsf{Re}(\lambda) < 0$, it follows that $\|e^{Gt}\| \to 0$ if and only if $\mathsf{Re}(\lambda) < 0$. This proves the first part.

Moreover, setting $C = \|P^{-1}|_2 \|P\|_2$, it follows that

$$\|e^{Gt}\|_2 \le C \|e^{\Lambda t}\|_2 = C \cdot \max_{\lambda \in \{\lambda_1, \dots, \lambda_n\}} |e^{\lambda t}|,$$

where the second factor remains bounded (by 1) as $t \to \infty$ if and only if $\mathsf{Re}(\lambda) \le 0$. This proves the second part, as the semi-simplicity condition is already contained in the diagonalizability assumption. ∎

To illustrate that the semi-simplicity assumption for critical eigenvalues with $\mathsf{Re}(\lambda) = 0$ is needed in Theorem 3.2, consider the matrix

$$G = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

which has an eigenvalue 0 that is *not* semi-simple. Then, by definition (3.2), we have

$$e^{Gt} = \begin{pmatrix} 1 & t \\ 0 & 1 \end{pmatrix},$$

which is clearly not bounded as $t \to \infty$.

We now consider the application of the explicit Euler method with step size h to (3.1):

$$\mathbf{y}_{i+1} = \mathbf{y}_i + hG\mathbf{y}_i = (I + hG)\mathbf{y}_i. \tag{3.3}$$

More generally, similar to the proof of Lemma 2.5 it can be shown that the application of an s-stage explicit Runga-Kutta method applied to (3.1) yields

$$\mathbf{y}_{i+1} = S(hG)\mathbf{y}_i \tag{3.4}$$

for a polynomial S of degree at most s.

Both, (3.3) and (3.4) are examples of a **linear discrete-time system** of the form

$$\mathbf{y}_{i+1} = D\mathbf{y}_i \tag{3.5}$$

for some matrix $D \in \mathbb{R}^{d\times d}$ and initial values $\mathbf{y}_0 \in \mathbb{R}^d$. Similar to IVPs, we can define a notion for the discrete case; (3.5) is called

- **asymptotically stable** if $\|\mathbf{y}_k\| \to 0$ as $k \to \infty$ for *all* initial values $\mathbf{y}_0 \in \mathbb{R}^d$;
- **stable** if there is a constant C (independent of t and $\mathbf{y}_0$) such that $\|\mathbf{y}_k\| < C\|\mathbf{y}_0\|$ holds for all $k \geq 0$ and $\mathbf{y}_0 \in \mathbb{R}^d$;
- **unstable**, otherwise.

Since the solution of (3.5) is clearly $\mathbf{y}_i = D^i\mathbf{y}_0$, the stability (3.5) is equivalently characterized by the growth of D^i. Analogous to Theorem 3.2, we have the following eigenvalue characterization.

Theorem 3.3 *The linear discrete-time system* (3.5) *is asymptotically stable if and only if all eigenvalues λ of D satisfy $|\lambda| < 1$.*

The linear discrete-time system (3.5) *is stable if and only if all eigenvalues λ of D satisfy $|\lambda| \leq 1$* and *if every eigenvalue λ with $|\lambda| = 1$ is semi-simple.*

3.1.1 Absolute stability

Any one-step method for approximating the solution of an IVP can be seen as a discrete-time system. The stability *of the method* is concerned with the following question:

Does the discrete-time system inherit the stability of the IVP?

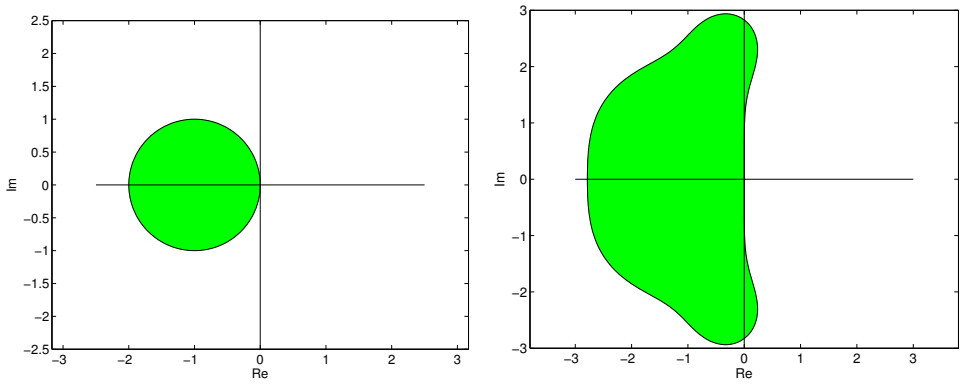


Figure 3.2. The green areas denote the stability regions for the forward Euler method (left plot) and the classical Runga-Kutta method (right plot).

We will study this question for the linear IVP (3.1). In this case, we have already seen that Runge-Kutta methods (and this holds for any linear one-step method) can be written as

$$\mathbf{y}_{i+1} = S(hG)\mathbf{y}_i.$$

for some function S, which is typically a polynomial (in the case of explicit Runge-Kutta methods) or a rational function (in the case of implicit Runge-Kutta methods defined below). The question above amounts to investigating whether the eigenvalues of $S(hG)$ have absolute magnitude less than 1. Note that the eigenvalues of $S(hG)$ are given by $S(h\lambda)$ for every eigenvalue λ of G. Let us therefore define the **stability domain** as

$$\mathcal{S} := \{z \in \mathbb{C} : \, |S(z)| \leq 1\}. \tag{3.6}$$

Assuming that the IVP is stabble, the method is also stable if

$$h\lambda \in \mathcal{S}$$

for every eigenvalue λ of G.[1]

Examples for stability regions are given in Figure 3.2. Note that we have

$$S(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$$

for the classical Runge-Kutta method (RK4). The best what can happen to is $h\lambda \in \mathcal{S}$ no matter how small (or large) h is chosen. Since we are only interested in $\mathsf{Re}(\lambda) \leq 0$, this ideal case is guaranteed to happen when $\mathbb{C}^- \subset \mathcal{S}$.

[1]Since S has to be analytic in a neighborhood of $\mathcal{S}$, an eigenvalue $S(h\lambda)$ can have modulus one only if $h\lambda \in \partial\mathcal{S}$. The fact that such eigenvalues are semi-simple follows from the fact that a matrix function does not change the block sizes of the Jordan normal form.

Definition 3.4 *A method is called A-stable if its stability region $\mathcal{S}$ satisfies $\mathbb{C}^- \subset \mathcal{S}$, where $\mathbb{C}^-$ denotes the left-half complex plane.*

Figure 3.2 clearly shows that neither the explicit Euler nor the classical Runge-Kutta methods are A-stable. More generally, we have the following negative result.

Lemma 3.5 *The stability domain $\mathcal{S}$ of any explicit Runga-Kutta method is compact.*

Proof. For an explicit Runga-Kutta method, the function S defining $\mathcal{S}$ in (3.6) is a polynomial different of degree at least 1. Since every such polynomial satisfies $|S(z)| \to \infty$ for $|z| \to \infty$, the stability domain must be bounded. ∎

A compact stability domain necessarily imposes a restriction on the step size. The step size for which the ray $h\lambda$ first leaves $\mathcal{S}$ is called **critical step size**. Under a mild additional assumption[2], the critical step size is given by

$$h_c = \inf\{h > 0 : h\lambda \notin \mathcal{S}\}. \tag{3.7}$$

Example 3.6 For the forward Euler method the stability region is a ball of radius 1 centered at -1. If G has only real negative eigenvalues, this implies that the critical step size is given by

$$h_c = \frac{2}{\max\{|\lambda| : \lambda \text{ is an eigenvalue of } G\}} \tag{3.8}$$

According to this analysis, the forward Euler method applied to Example 3.1 yields the desired asymptotic behavior if $|1 - 500\,h| \lesssim 1$, that is, $h \lesssim 1/250$. The obtained approximation for $h = 1/100$ is displayed in Figure 3.3; it "explodes" after a short time, demonstrating very clearly the risk of choosing h too large. For $h = 1/200$, this explosion is avoided, but the correct asymptotic behavior is only reflected for $h = 1/250$ or smaller. ◇

The compactness of the stability region imposes severe restrictions on the step size for discretizations of parabolic PDEs. As an illustration, consider the ordinary differential equation

$$\cdot\mathbf{y}(t) = G\mathbf{y}(t), \quad \text{with} \quad G = \frac{1}{h_x^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 2 \end{pmatrix},$$

[2]There is the possibility that the ray touches the boundary of the stability domain before, at some $h_- < h_c$. In that case $S(h_-)$ has at least one eigenvalue of magnitude 1. If this eigenvalue is not semi-simple then the critical step size is h_-. For simplicity, we do not consider this pathological case.

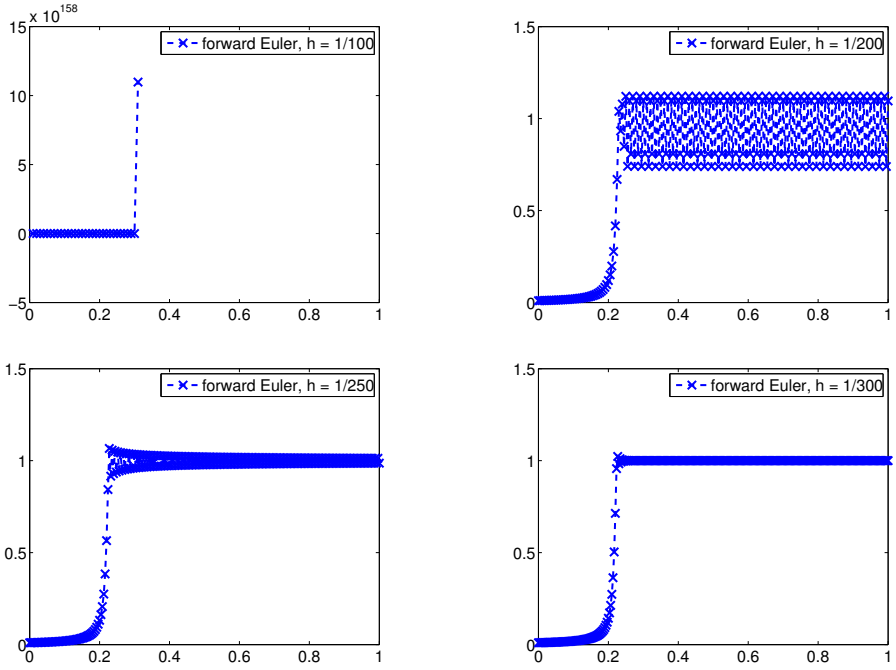


Figure 3.3. Result of forward Euler applied to Example 3.1 with different step sizes h.

which arises from the central finite-difference discretization with mesh width h_x of the one-dimensional parabolic PDE

$$\begin{aligned} &\frac{\partial}{\partial t}u(t,x) = \frac{\partial^2}{\partial x^2}u(t,x), \qquad x \in]0,1[\\ &u(t,0) = u(t,1) = 0. \end{aligned}$$

There are explicit formulas for the eigenvalues of G, which imply that all eigenvalues are real and negative. Moreover, the smallest eigenvalue satisfies $|\lambda| \sim h_x^{-2}$ Hence, according to (3.8), the critical step size of the forward Euler method applied to this IVP satisfies $h_c \sim h_x^2$. This (highly undesirable) condition is typical when applying explicit methods to discretizations of parabolic PDEs.

3.1.2 The implicit Euler method

Since explicit methods always yield polynomial stability function, there is no hope to obtain an explicit A-stable method. In this section, we will discuss the most basic implicit method, the **implicit Euler method**:

$$\mathbf{y}_1 = \mathbf{y}_0 + hf(t_1, \mathbf{y}_1).$$

In contrast to explicit methods, we need to solve a system of nonlinear equations to determine $\mathbf{y}_1$! This should be done by the Newton method (or some simplified variant), see Section 3.2.1.

The stability function of the implicit Euler method is given by

$$S(z) = \frac{1}{1-z},$$

and hence the stability region is the complement of a disc in the complex plane, see Figure 3.4.

3.1.3 Beyond linear ODEs*

Let a general *autonomous* IVP

$$\dot{\mathbf{y}}(t) = f\big(\mathbf{y}(t)\big), \qquad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{3.9}$$

have a stationary point $\mathbf{y}^\star$ and assume that $\mathbf{y}(t) \to \mathbf{y}^\star$ as $t \to \infty$ for every $\mathbf{y}_0$ sufficiently close to $\mathbf{y}^\star$. Then the linearization around this stationary point takes the form

$$\dot{\mathbf{y}}(t) = f'(\mathbf{u}^\star) \cdot \mathbf{y}(t), \qquad \mathbf{y}(t_0) = \mathbf{y}_0,$$

where all eigenvalues of the derivative $f'(\mathbf{u}^\star) \in \mathbb{R}^{d\times d}$ have negative real part. A Runge-Kutta method with step size h applied to (3.9) converges to $\mathbf{y}^\star$ for sufficiently close initial values $\mathbf{y}_0$ if λh is in the stability region for every eigenvalue λ of $f'(\mathbf{u}^\star)$.

A number of crimes are committed when considering the linearization only. The concept of B-stability is better suited for nonlinear ODEs; see Chapter IV.12 in [HW].

3.2 General form of implicit Runge-Kutta methods

The implicit Euler method discussed above belongs to a whole class of implicit Runge-Kutta methods. These are obtained as a generalization of the explicit Runge-Kutta methods, by giving up the requirement that we can compute the stages by means of forward substitution.

Definition 3.7 *Suppose that* $\mathbf{k}_1, \dots, \mathbf{k}_s \in \mathbb{R}^d$ *satisfy the following nonlinear equations:*

$$\begin{aligned}
\mathbf{k}_1 &= f(t_0 + c_1 h, \mathbf{y}_0 + h \sum_{\ell=1}^{s} a_{1\ell} \mathbf{k}_\ell) \\
&\vdots \\
\mathbf{k}_s &= f\Big(t_0 + c_s h, \mathbf{y}_0 + h \sum_{\ell=1}^{s} a_{s\ell} \mathbf{k}_\ell\Big),
\end{aligned}$$

for given coefficients $a_{i\ell}, c_i \in \mathbb{R}$. *Then*

$$\mathbf{y}_1 = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i \mathbf{k}_i,$$

is one step of the s**-stage implicit Runge-Kutta method**.

Again, the coefficients defining an implicit Runge-Kutta method can be organized compactly in a Butcher tableau:

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array} := \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1,s} \\ c_2 & a_{21} & \cdots & a_{2,s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

Note that in contrast to explicit methods, the matrix A is now a general matrix and not required to be strictly lower triangular. The three simplest examples of implicit Runge-Kutta methods:

implicit Euler method

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

implicit midpoint rule

$$\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

implicit trapezoidal rule

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline 0 & 1/2 & 1/2 \end{array}$$

The implicit trapezoidal rule has a particularly nice stability region, see Figure 3.4.

For Definition 3.7 to make sense, we need to study the solvability of the system of nonlinear equations defining the stages. Moreover, some uniqueness property of the solutions should be imposed to yield a sensible step. For this purpose, we will reformulate the system as a system of fixed point equations by introducing the quantities

$$\mathbf{g}_i = \mathbf{y}_0 + h\sum_{\ell=1}^{s} a_{i\ell}\mathbf{k}_\ell, \quad i = 1, \ldots, s.$$

Then the solutions $\mathbf{g}_1, \ldots, \mathbf{g}_s$ of the system of fixed point equations

$$\mathbf{g}_i = \mathbf{y}_0 + h\sum_{\ell=1}^{s} a_{i\ell} f(t_0 + c_\ell h, \mathbf{g}_\ell), \quad i = 1, \ldots, s, \tag{3.10}$$

define the next step as

$$\mathbf{y}_1 = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i f(t_0 + c_i h, \mathbf{g}_i), \tag{3.11}$$

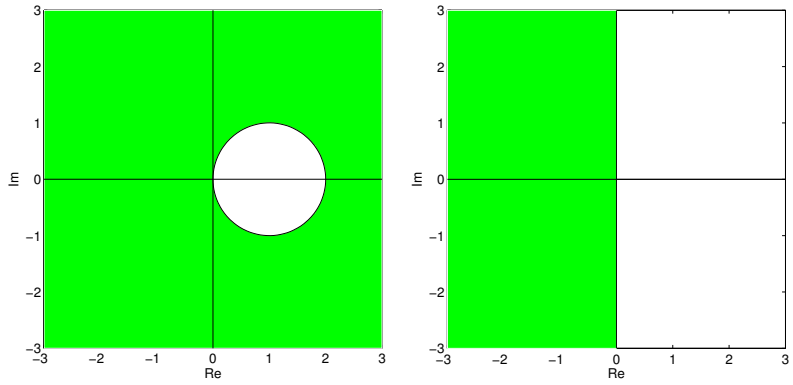


Figure 3.4. Stability regions for the implicit Euler method (left plot) and the implicit trapezoidal rule (right plot).

This is clearly equivalent to Definition 3.7, as can be seen from the relation

$$\mathbf{k}_i = f(t_0 + c_i h, \mathbf{g}_i), \quad i = 1, \ldots, s.$$

Theorem 3.8 *Let $f \in C(\Omega, \mathbb{R}^d)$ be Lipschitz continuous with respect to the state on the augmented phase space $\Omega \subset \mathbb{R} \times \mathbb{R}^d$. Then there exists $h_* > 0$ and unique functions $\mathbf{g}_i \in C(]-h_*, h_*[, \mathbb{R}^d)$, such that*

1. *$\mathbf{g}_i(0) = \mathbf{y}_0$ for $i = 1, \ldots, s$;*
2. *for all $0 \le h < h_*$, the vectors $\mathbf{g}_i(h)$ satisfy the equations (3.10).*

Proof. We only give a sketch of the proof and refer to Theorem 6.28 in [DB] for details. Let us rewrite the system (3.10) as a single equation

$$\mathbf{g} = F(\mathbf{g}), \text{ with } \mathbf{g} = \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_s \end{pmatrix}, \quad F(\mathbf{g}) = \begin{pmatrix} \mathbf{y}_0 + h\sum_{\ell=1}^s a_{1,\ell} f(t_0 + c_\ell h, \mathbf{g}_\ell) \\ \vdots \\ \mathbf{y}_0 + h\sum_{\ell=1}^s a_{s\ell} f(t_0 + c_\ell h, \mathbf{g}_\ell) \end{pmatrix}. \tag{3.12}$$

Then

$$\begin{aligned} \|F(\widehat{\mathbf{g}}) - F(\mathbf{g})\|_\infty &\le h\|A\|_\infty \max_{1\le\ell\le s} \|f(t_0 + c_\ell h, \widehat{\mathbf{g}}_\ell) - f(t_0 + c_\ell h, \mathbf{g}_\ell)\|_\infty \\ &\le h\|A\|_\infty L \|\widehat{\mathbf{g}} - \mathbf{g}\|_\infty, \end{aligned}$$

where we used the Lipschitz continuity of f (with Lipschitz constant L). Hence, F is a contraction provided that

$$h < h_* := \frac{1}{\|A\|_\infty L}. \tag{3.13}$$

By the Banach fixed point theorem, this shows the solvability (3.10). To show the continuous dependence of the solution $\mathbf{g}$ on h (and its uniqueness) requires the application of a parameter-dependent fixed point theorem, which can be found, e.g., in [Dieudonné, J. Foundations of Modern Analysis. 1960]. □

It is interesting to discuss the condition (3.13) on the step size h for a linear ODE $\dot{\mathbf{y}} = G\mathbf{y}$. Then a suitable Lipschitz constant is given by $L = \|G\|_\infty$. Hence, (3.13) becomes $h < \frac{1}{\|A\|_\infty \|G\|_\infty}$, which appears to be a restriction on the step size not better than the restrictions for explicit methods to be stable! However, this only shows that *fixed point iterations are unsuitable* for solving the nonlinear system defining the stages. For solving the nonlinear system, other methods like the Newton method should be used, see also Section 3.2.1.

If we additionally require $f \in C^p(\Omega, \mathbb{R}^d)$ for some $p \geq 1$ in Theorem 3.8 then it can be shown that the vectors $\mathbf{g}_i$ (and therefore also the step $\mathbf{y}_1$) are p times continuously differentiable functions in h. This allows to carry over the discussion of Section 2.2.1 on order conditions to implicit Runge-Kutta methods in a nearly verbatim manner. In particular, an implicit Runge-Kutta method is consistent for all $f \in C(\Omega, \mathbb{R}^d)$ if and only if

$$\mathbf{b}^\mathsf{T}\mathbf{e} = 1.$$

It is invariant under under autonomization if and only if it is consistent and satisfies

$$\mathbf{c} = A\mathbf{e}.$$

The order conditions of Theorem 2.12 also apply, only that $A^{(\beta)}$ is now defined for a general matrix A. Table 2.1 is still valid. The biggest difference is that we now have more coefficients available to design the method and potentially achieve higher order. In fact, the order can be larger than s (but not larger than $2s$). For example, the implicit midpoint rule has order 2.

Finally, we give a compact formula for the stability function of an implicit Runge-Kutta method.

Lemma 3.9 *The stability function of an s-stage implicit Runge-Kutta method is given by*

$$S(z) = 1 + zb^\mathsf{T}(I - zA)^{-1}\mathbf{e},$$

which is a rational function.

Proof. When applying the implicit Runge-Kutta method to the linear IVP $\dot{y}(t) = \lambda y(t)$ we obtain from (3.12) the linear system

$$\mathbf{g} = y_0\mathbf{e} + h\lambda A\mathbf{g}$$

and hence $\mathbf{g} = y_0(I - h\lambda A)^{-1}\mathbf{e}$. According to (3.11) the next step is given by

$$y_1 = y_0 + y_0 h\lambda b^\mathsf{T}(I - h\lambda A)^{-1}\mathbf{e} = \underbrace{(1 + h\lambda b^\mathsf{T}(I - h\lambda A)^{-1}\mathbf{e})}_{=S(h\lambda)} y_0.$$

This shows the first statement. The second statement follows from the fact that the entries of $(I - zA)^{-1}$ are rational functions in z, which is a consequence of $(I - zA)^{-1} = \operatorname{adj}(I - zA)/\det(I - zA)$. □

3.2.1 Solution of the nonlinear system defining the stages

The implementation of an implicit Runge-Kutta method requires the solution of the nonlinear equations (3.10). Since $\mathbf{g}_i - \mathbf{y}_0 = O(h)$ there is the risk of numerical cancellation for small step sizes h. It is therefore preferable to work with the smaller quantities

$$\mathbf{z}_i := \mathbf{g}_i - \mathbf{y}_0.$$

Then (3.10) becomes

$$\mathbf{z}_i = h \sum_{\ell=1}^{s} a_{i\ell} f(t_0 + c_\ell h, \mathbf{y}_0 + \mathbf{z}_\ell), \quad i = 1, \ldots, s,$$

which can be written as

$$\mathbf{z} := \begin{pmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_s \end{pmatrix} = (A \otimes I) \begin{pmatrix} hf(t_0 + c_1 h, \mathbf{y}_0 + \mathbf{z}_1) \\ \vdots \\ hf(t_0 + c_s h, \mathbf{y}_0 + \mathbf{z}_s) \end{pmatrix} =: (A \otimes I) F(\mathbf{z}), \tag{3.14}$$

where $\otimes$ denotes the Kronecker product between two matrices. Once $\mathbf{z}_1, \ldots, \mathbf{z}_s$ are determined, we could then determine the next step by

$$\mathbf{y}_1 = \mathbf{y}_0 + h \sum_{i=1}^{s} b_i f(t_0 + c_i h, \mathbf{y}_0 + \mathbf{z}_i),$$

This seems to suggest that s additional function evaluations are needed. In fact this can be avoided with a small trick. Assuming that A is invertible, it follows from (3.14) that we can write

$$\mathbf{y}_1 = \mathbf{y}_0 + \sum_{i=1}^{s} d_i \mathbf{z}_i,$$

where

$$\begin{pmatrix} d_1, & \ldots, & d_s \end{pmatrix} := \mathbf{b}^T A^{-1}.$$

The kth step of the **Newton method applied to the nonlinear system** (3.14) takes the following form:

$$\begin{array}{rl} \text{Solve linear system} & \big(I - (A \otimes I) F'(\mathbf{z}^k)\big) \triangle \mathbf{z}^k = -\mathbf{z}^k + h(A \otimes I) F(\mathbf{z}^k), \\ \text{Update} & \mathbf{z}^{k+1} = \mathbf{z}^k + \triangle \mathbf{z}^k. \end{array} \tag{3.15}$$

Note that each step of (3.15) requires to solve a linear system. In practice, this is usually done via computing a (sparse) LU factorization of the matrix

$$\begin{aligned} & I - (A \otimes I) F'(\mathbf{z}^k) \\ =& I - h \begin{pmatrix} a_{11} \frac{\partial f}{\partial \mathbf{y}}(t_0 + c_1 h, \mathbf{y}_0 + \mathbf{z}_1^k) & \cdots & a_{1s} \frac{\partial f}{\partial \mathbf{y}}(t_0 + c_1 h, \mathbf{y}_0 + \mathbf{z}_s^k) \\ \vdots & & \vdots \\ a_{s1} \frac{\partial f}{\partial \mathbf{y}}(t_0 + c_s h, \mathbf{y}_0 + \mathbf{z}_1^k) & \cdots & a_{ss} \frac{\partial f}{\partial \mathbf{y}}(t_0 + c_s h, \mathbf{y}_0 + \mathbf{z}_s^k) \end{pmatrix}. \end{aligned}$$

The factorization of this $sd \times sd$ matrix is often the (by far) most expensive part and needs to be performed in every step of the Newton method (3.15).

We can reduce the cost significantly by, replacing all Jacobians $\frac{\partial f}{\partial \mathbf{y}}(t_0 + c_1 h, \mathbf{y}_0 + \mathbf{z}_s^k)$ with an approximation

$$J \approx \frac{\partial f}{\partial \mathbf{y}}(t_0, \mathbf{y}_0).$$

The resulting **simplified Newton method** takes the form:

$$\begin{array}{rl} \text{Solve linear system} & \big(I - (A \otimes I)J\big) \triangle \mathbf{z}^k = -\mathbf{z}^k + h(A \otimes I)F(\mathbf{z}^k), \\ \text{Update} & \mathbf{z}^{k+1} = \mathbf{z}^k + \triangle \mathbf{z}^k. \end{array} \tag{3.16}$$

Now, the LU factorization of $I - (A \otimes I)J$ needs to be computed only once and can then be reused. Hence, we only need to perform 1 LU factorization in each step of the implicit Runge-Kutta method.

The choice

$$\mathbf{z}^0 = \mathbf{0}$$

usually represents a very good starting value, since the exact solution is known to satisfy $\|\mathbf{z}\| = O(h)$. Suggestions for better starting values can be found in Section IV.8 of [HW], which also discusses a suitable stopping criterion for (3.16).

3.2.2 Examples of implicit Runge-Kutta methods

As described in [DB] and [HW], collocation combined with numerical quadrature provides a way to construct high-order implicit Runge-Kutta methods.

Gauss methods. Gauss quadrature yields s-stage implicit Runge-Kutta methods that are A-stable and have order $2s$ (which is optimal). For $s = 1$, the Gauss method coincides with the implicit midpoint rule. For $s = 2$, the Gauss method has the following Butcher tableau:

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Radau methods. Radau quadrature yields s-stage implicit Runge-Kutta methods that are A-stable and have order $2s-1$. Moreover, Radau methods have a property called L-stability, see Section 3.4 below. For $s = 1$, the Radau method is our good old friend – the implicit Euler method. For $s = 2$ and $s = 3$, the Radau methods have the following Butcher tableaus:

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{9} & \frac{-1-\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{6-\sqrt{6}}{10} & \frac{1}{9} & \frac{88+7\sqrt{6}}{360} & \frac{88-43\sqrt{6}}{360} \\ \frac{6+\sqrt{6}}{10} & \frac{1}{9} & \frac{88+43\sqrt{6}}{360} & \frac{88-7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{16+\sqrt{6}}{36} & \frac{16-\sqrt{6}}{36} \end{array}$$

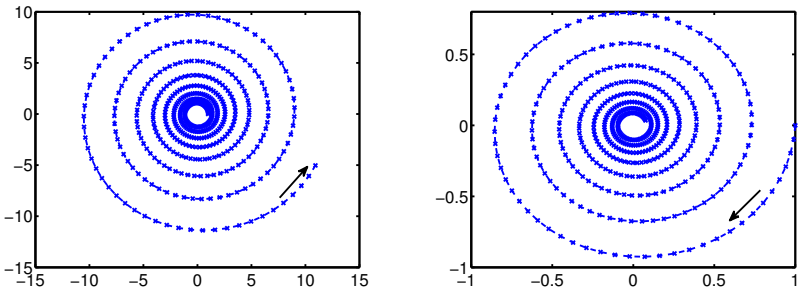


Figure 3.5. *Forward Euler (left plot) and implicit Euler (right plot) applied to (3.17).*

RADAU5, one of the most popular codes for solving DAEs, is based on the 3-stage Radau method.

3.3 The danger of being too stable

Despite all the praise for A-stable methods, they actually bear the danger of turning an unstable or a non-asymptotically stable IVP into an asymptotically stable discrete-time system. This point is nicely illustrated with

$$\dot{\mathbf{y}}(t) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{y}(t), \quad \mathbf{y}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} =: \mathbf{y}_0, \tag{3.17}$$

which arises, e.g., from the model of a spring pendulum. The solution stays on the unit circle and does not converge to a stationary point. Figure 3.5 displays the approximations on the interval $[0, 10]$ obtained from the forward/implicit Euler methods with step size $h = 1/10$. It can be seen that the forward Euler method does not stay on the unit circle and the solution drifts away as t increases. This is not unexpected: The eigenvalues of the matrix $A = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}$ are $\lambda = \pm\omega\mathrm{i}$. Hence, no matter how small h is, $h\lambda$ is not in the stability region of the forward Euler method. Unfortunately, the behavior of the implicit Euler method is not much better. The solution does also not stay on the unit circle and approaches zero for long times. Ironically, the problem is now that $h\lambda$ is in the interior of the stability region of the implicit Euler method. Thus, the approximate solution always converges to zero, no matter how small h is.

To avoid the phenomena described above, we need a method for which $h\lambda$ is on the boundary of the stability region. Such a method is given by the trapezoidal method, see Figure 3.4. Indeed, this method produces the exact asymptotic behavior even for relatively large h, see Figure 3.6.

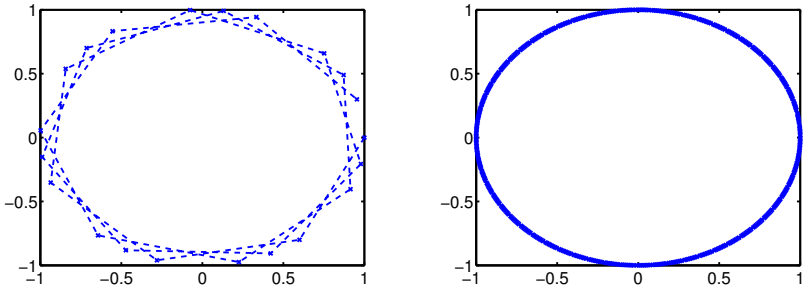


Figure 3.6. *Implicit trapezoidal method with $h = 1$ (left plot) and $h = 0.1$ (right plot) applied to (3.17).*

3.4 L-stability

Section 3.3 may misleadingly indicate that it is *always* desirable to have the stability region coincide with the left half-plane. In fact, for very stiff problems this is not desirable at all. For a rational stability function S, we have

$$\lim_{x\to-\infty} S(x) = \lim_{x\to\infty} S(x) = \lim_{y\to\infty} S(\mathrm{i}y).$$

If the stability region coincide with the left half-plane, then $|S(\mathrm{i}y)| = 1$ for all $y \in \mathbb{R}$. In turn, $|S(z)|$ is close to 1 whenever z is close to the real axis and has large negative real part. In practice, this means that such a method damps errors only very slowly for stiff problems. To see this, let us consider the IVP

$$\dot{y}(t) = -2000(y(t) - \cos t), \qquad y(0) = 0. \tag{3.18}$$

As can be seen from Figure 3.7, the initial numerical oscillation is damped much more quickly for the implicit Euler method compared to the trapezoidal method. This is due to the following property.

Definition 3.10 *A method is called L-stable if it is A-stable and if, additionally,*

$$\lim_{x\to\infty} R(x) = 0.$$

For an implicit Runge-Kutta method defined by $\mathbf{b}, \mathbf{c}, A$ with nonsingular A, we have

$$S(\infty) = 1 - \mathbf{b}^T A^{-1}\mathbf{e},$$

where $\mathbf{e}$ is the vector of all ones. Hence, the method is L-stable if and only if $\mathbf{b}^T A^{-1}\mathbf{e} = 1$. Examples of L-stable methods include the implicit Euler method and the Radau methods mentioned above.

Apart from stiff ODEs, L-stability also plays an important role when solving differential-algebraic equations (DAEs), that is, ODEs with additional algebraic side constraints.

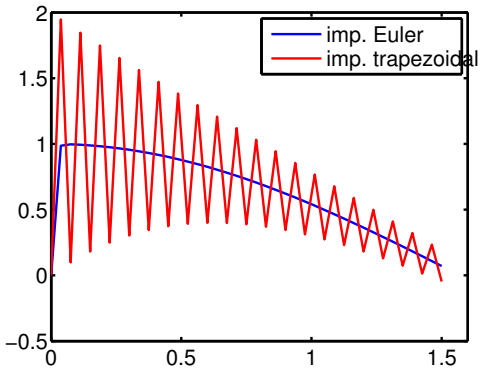


Figure 3.7. *Implicit Euler and implicit trapezoidal method with* $h = 1.5/40$ *applied to (3.18).*

3.5 Rosenbrock methods*

When using the simplified Newton method (3.16), one has to strike a balance between the stopping criterion the inner iterations and h. It would be much simpler if we could just use one iteration and let the accuracy be handled solely by the step size control for the Runge-Kutta method. **Rosenbrock methods** (also called linearly implicit Runge-Kutta methods) rationalize this idea. They can be motivated by considering an implicit Runge-Kutta method with a lower triangular matrix A applied to an autonomous ODE:

$$\mathbf{k}_i = f\Big(\mathbf{y}_0 + h\sum_{\ell=1}^{i-1} a_{i\ell}\mathbf{k}_\ell + a_{ii}\mathbf{k}_i\Big), \quad i = 1,\dots,s,$$
$$\mathbf{y}_1 = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i\mathbf{k}_i.$$

Such a method is also called DIRK (diagonally implicit Runge-Kutta method). Linearizing this formula in order to get rid off the implicit part yields

$$\mathbf{k}_i = f(\mathbf{g}_i) + hf'(\mathbf{g}_i)a_{ii}\mathbf{k}_i,$$

where

$$\mathbf{g}_i = \mathbf{y}_0 + h\sum_{\ell=1}^{i-1} a_{i\ell}\mathbf{k}_\ell$$

for $i = 1,\dots,s$. This is one step of the Newton method. Now we replace $f'(\mathbf{g}_i)$ by $J = f'(\mathbf{y}_0)$ and obtain one step of the simplified Newton method. Some additional freedom is gained by allowing for linear combinations of $J\mathbf{k}_\ell$.

Definition 3.11 *An s-stage Rosenbrock method is defined by*

$$\mathbf{k}_i = f\Big(\mathbf{y}_0 + h\sum_{\ell=1}^{i-1} a_{i\ell}\mathbf{k}_\ell\Big) + hJ\sum_{\ell=1}^{i}\gamma_{i\ell}\mathbf{k}_\ell, \quad i = 1,\dots,s,$$
$$\mathbf{y}_1 = \mathbf{y}_0 + h\sum_{i=1}^{s} b_i\mathbf{k}_i,$$

with coefficients $\alpha_{i\ell}, \gamma_{i\ell}, b_i$ *and* $J = f'(\mathbf{y}_0)$.

Note that each stage of the Rosenbrock method requires to solve a linear system with the matrix $I - h\gamma_{ii}J$. Naturally, methods with $\gamma_{11} = \cdots = \gamma_{ss} \equiv \gamma$ are of particular interest, because they allow to reuse the LU factorization of $I - h\gamma J$ when having to solve the linear system for each stage. We refer to Section IV.7 in [HW] for the construction of specific Rosenbrock methods.

A particularly simple example of a Rosenbrock method is given by:

$$(I - h\gamma J)\mathbf{k}_1 = f(\mathbf{y}_0), \qquad \gamma = \frac{1}{2+\sqrt{2}},$$
$$(I - h\gamma J)\mathbf{k}_2 = f\Big(\mathbf{y}_0 + \frac{1}{2}h\mathbf{k}_1\Big) - h\gamma J\mathbf{k}_1$$
$$\mathbf{y}_1 = \mathbf{y}_1 + h\mathbf{k}_2.$$

A modified variant of this second-order method is behind MATLAB's `ode23s`; see [L. F. Shampine and M. W. Reichelt: The MATLAB ODE suite] for more details.

Part II

Optimization

Chapter 4

Unconstrained optimization

An **unconstrained optimization problem** takes the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{4.1}$$

for a **target functional** (also called **objective function**) $f : \mathbb{R}^n \to \mathbb{R}$. In this chapter and throughout most of our discussion on optimization, we will assume that f is sufficiently smooth, that is, at least continuously differentiable.

In most applications of optimization problem, one is usually interested in a **global minimizer** $\mathbf{x}^*$, which satisfies $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all x in $\mathbb{R}^n$ (or at least for all x in the domain of interest). Unless f is particularly nice, optimization algorithms are often not guaranteed to yield global minima but only yield local minima. A point $\mathbf{x}^*$ is called a **local minimizer** if there is a neighborhood $\mathcal{N}$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}$. Similarly, $\mathbf{x}^*$ is called a **strict local minimizer** $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}$ with $\mathbf{x} \neq \mathbf{x}^*$.

4.1 Fundamentals

Sufficient and necessary conditions for local minimizers can be developed from the Taylor expansion of f. Let us recall Example 2.3: If f is two times continuously differentiable then

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T\mathbf{h} + \frac{1}{2}\mathbf{h}^\mathsf{T} H(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^3), \tag{4.2}$$

where ∇f is the gradient and $H = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)_{i,j=1}^m$ is the Hessian *[matrice Hessienne]* of f.

Theorem 4.1 (First-order necessary condition) *If $\mathbf{x}^*$ is a local minimizer and f is continuously differentiable in an open neighborhood of $\mathbf{x}^*$ then $\nabla f(\mathbf{x}^*) = 0$.*

Proof. By local optimality $[f(\mathbf{x}^* + t\mathbf{d}) - f(\mathbf{x}^*)]/t$ is nonnegative for sufficiently small $t > 0$ and for arbitrary $\mathbf{d}$. It follows that

$$\lim_{t \to 0^+} \frac{1}{t}[f(\mathbf{x}^* + t\mathbf{d}) - f(\mathbf{x}^*)] = \nabla f(\mathbf{x}^*)^T \mathbf{d} \geq 0.$$

Choosing $\mathbf{d} = -\nabla f(\mathbf{x}^*)$ implies $\nabla f(\mathbf{x}^*) = 0$. □

A $\mathbf{x}^*$ satisfying $\nabla f(\mathbf{x}^*) = 0$ is called **stationary point**. A **saddle point** is a stationary point that is neither a local minimizer nor a local maximizer. More can be said if the Hessian of f is available.

Theorem 4.2 (Second-order necessary condition) *If $\mathbf{x}^*$ is a local minimizer and f is two times continuously differentiable in an open neighborhood of $\mathbf{x}^*$ then $\nabla f(\mathbf{x}^*) = 0$ and the Hessian $H(\mathbf{x}^*)$ is positive semidefinite.*

Proof. Theorem 4.1 already yields $\nabla f(\mathbf{x}^*) = 0$, so it remains to prove the positive semidefiniteness of $H(\mathbf{x}^*)$. For sufficiently small t and arbitrary $\mathbf{d}$, we have

$$\begin{aligned} 0 \leq f(\mathbf{x}^* + t\mathbf{d}) - f(\mathbf{x}^*) &= t\nabla f(\mathbf{x}^*)^T \mathbf{d} + \frac{t^2}{2}\mathbf{d}^\mathsf{T} H(\mathbf{x}^*)\mathbf{d} + O(t^3) \\ &= \frac{t^2}{2}\mathbf{d}^\mathsf{T} H(\mathbf{x}^*)\mathbf{d} + O(t^3), \end{aligned}$$

where we used the Taylor expansion (4.2). Hence, $\mathbf{d}^\mathsf{T} H(\mathbf{x}^*)\mathbf{d} \geq O(t)$ and the result follows by taking the limit $t \to 0$. □

Theorem 4.3 (Second-order sufficient condition) *Suppose that f is two times continuously differentiable in an open neighborhood of $\mathbf{x}^*$ and that $\nabla f(\mathbf{x}^*) = 0$. Moreover, suppose that the Hessian $H(\mathbf{x}^*)$ is positive definite. Then $\mathbf{x}^*$ is a strict local minimizer.*

Proof. Since $H(\mathbf{x}^*)$ is positive definite, there is a constant μ such that

$$\mathbf{d}^T H(\mathbf{x}^*)\mathbf{d} \geq \mu \|\mathbf{d}\|_2^2$$

for all $\mathbf{d} \in \mathbb{R}^n$. Using the Taylor expansion and $\nabla f(\mathbf{x}^*) = 0$, we have

$$f(\mathbf{x}^* + \mathbf{d}) - f(\mathbf{x}^*) = \frac{1}{2}\mathbf{d}^\mathsf{T} H(\mathbf{x}^*)\mathbf{d} + O(\|\mathbf{d}\|^3) \geq \frac{\mu}{2}\|\mathbf{d}\|_2^2 + O(\|\mathbf{d}\|^3) \geq \frac{\mu}{4}\|\mathbf{d}\|_2^2 > 0$$

for all $\mathbf{d} \neq 0$ of sufficiently small norm. Hence, $\mathbf{x}^*$ is a strict local minimizer. □

4.2 Line search methods

General line search methods for solving the optimization problem (4.1) take the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{4.3}$$

where $\alpha_k > 0$ is called the **step length** and $\mathbf{p}_k$ is called the **search direction**.

As we will see, there are many choices for α and $\mathbf{p}$. A natural requirement is that $\mathbf{p}$ should be chosen such that the slope of f in the direction $\mathbf{p}$ is negative. Because of

$$\lim_{t\to 0^+} \frac{f(\mathbf{x}+t\mathbf{p}) - f(\mathbf{x})}{\|t\mathbf{p}\|_2} = \nabla f(\mathbf{x})^T \mathbf{p},$$

this motivates the following definition.

Definition 4.4 *A vector $\mathbf{p} \neq 0$ is called* **descent direction** *of a continuously differentiable function f at a point $\mathbf{x}$ if $\nabla f(\mathbf{x})^T \mathbf{p} < 0$.*

4.2.1 Method of steepest descent

It makes sense to choose $\mathbf{p}$ such that the slope of f in the direction $\mathbf{p}$ is as small as possible. This leads to the minimization problem

$$\min_{\|\mathbf{p}\|_2=1} \nabla f(\mathbf{x})^T \mathbf{p}, \tag{4.4}$$

which can be easily solved.

Lemma 4.5 *If $\nabla f(\mathbf{x}) \neq 0$ then (4.4) has the unique solution*

$$\mathbf{p} = -\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2}.$$

Proof. By the Cauchy-Schwarz inequality we have

$$\nabla f(\mathbf{x})^T \mathbf{p} \geq -\|\nabla f(\mathbf{x})\|_2 \|\mathbf{p}\|_2 = -\|\nabla f(\mathbf{x})\|_2,$$

with equality if and only if $\mathbf{p}$ takes the form (4.4). □

Any vector of the form

$$\mathbf{p} = -\alpha \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2}, \quad \alpha > 0,$$

is called **direction of steepest descent**. It remains to choose the step length α_k.

The **Armijo rule** applies to a general line search method (4.3) and proceeds as follows: Let $\beta \in]0,1[$ (typically $\beta = 1/2$) and $c_1 \in]0,1[$ (for example $c_1 = 10^{-4}$) be fixed parameters.

Armijo rule:
Determine the largest number $\alpha_k \in \{1, \beta, \beta^2, \beta^3, \ldots\}$ such that

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) - f(\mathbf{x}_k) \leq c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \tag{4.5}$$

holds.

In words, the condition (4.5) ensures that the reduction in f is proportional to the step length and the directional derivative. The following lemma guarantees that (4.5) can always be satisfied provided that $\mathbf{p}_k$ is a descent direction.

Lemma 4.6 *Let $c_1 \in]0,1[$ and let $f : U \to \mathbb{R}$ be continuously differentiable in an open set $U \subset \mathbb{R}^n$. If $\mathbf{x} \in U$ and if $\mathbf{p}$ is a descent direction of f at $\mathbf{x}$ then there is $\overline{\alpha} > 0$ such that*

$$f(\mathbf{x} + \alpha\mathbf{p}) - f(\mathbf{x}) \leq c_1 \alpha \nabla f(\mathbf{x})^T \mathbf{p} \quad \forall \alpha \in [0, \overline{\alpha}].$$

Proof. The inequality trivially holds for $\alpha = 0$. Now, let $\alpha > 0$. Then

$$\frac{f(\mathbf{x} + \alpha\mathbf{p}) - f(\mathbf{x})}{\alpha} - c_1 \nabla f(\mathbf{x})^T \mathbf{p} \overset{\alpha \to 0^+}{\longrightarrow} \nabla f(\mathbf{x})^T \mathbf{p} - c_1 \nabla f(\mathbf{x})^T \mathbf{p} = (1 - c_1) \nabla f(\mathbf{x})^T \mathbf{p} < 0.$$

Hence, by choosing $\overline{\alpha}$ sufficiently small, we have

$$\frac{f(\mathbf{x} + \alpha\mathbf{p}) - f(\mathbf{x})}{\alpha} - c_1 \nabla f(\mathbf{x})^T \mathbf{p} \leq 0 \quad \forall \alpha \in [0, \overline{\alpha}].$$

This shows the result. □

In summary, we obtain Algorithm 4.7.

Algorithm 4.7 Steepest descent with Armijo line search
Input: *Function f, starting vector $\mathbf{x}_0$ and parameters $\beta > 0, c_1 > 0$. Tolerance* tol.
Output: *Vector $\mathbf{x}_k$ approximating stationary point.*

1: ***for*** $k = 0, 1, 2, \ldots$ ***do***
2: *Set* $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.
3: *Stop if* $\|\mathbf{p}_k\| \leq$ tol.
4: *Determine step length α_k according to the Armijo rule* (4.5).
5: *Set* $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.
6: ***end for***

4.2.2 Convergence of general line search methods

In the following, we will analyse the convergence of general line search method (4.3). Of course, we cannot choose arbitrary $\alpha_k, \mathbf{p}_k$ and still expect convergence. First of all, we would like to maintain the **Armijo condition** of Lemma 4.6:

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) - f(\mathbf{x}_k) \leq c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \tag{4.6}$$

for some $c_1 \in]0,1[$. This ensures a sufficient decrease in the objective function.

The condition (4.6) is not enough by itself to guarantee reasonable progress of the linear search method. Lemma 4.6 shows that it is always satisfied for sufficiently small α_k, provided that $\mathbf{p}_k$ is a descent direction, but we have to make sure that α_k does not become unacceptably small. This is the purpose of the so called **curvature condition**

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k \geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \tag{4.7}$$

which should hold for some $c_2 \in]c_1, 1[$. Typical values for c_2 are 0.9 (when $\mathbf{p}_k$ is chosen by a Newton or quasi-Newton method) or 0.1 (when $\mathbf{p}_k$ is chosen by the nonlinear conjugate gradient method).

The two conditions (4.6)–(4.7) taken together are called *Wolfe conditions*. The following lemma shows that there always exist step lengths satisfying the Wolfe conditions under reasonable assumptions on f.

Lemma 4.8 *Let f be continuously differentiable and assume that is bounded from below along the ray $\{\mathbf{x}_k + \alpha\mathbf{p}_k \mid \alpha > 0\}$ for some descent direction $\mathbf{p}_k$. Then there exist intervals of step lengths satisfying the Wolfe conditions* (4.6)–(4.7)*, provided that $0 < c_1 < c_2 < 1$.*

Proof. Let us consider the function

$$\psi(\alpha) = f(\mathbf{x}_k + \alpha\mathbf{p}_k) - f(\mathbf{x}_k) - c_1\alpha\nabla f(\mathbf{x}_k)^T\mathbf{p}_k.$$

Since $\mathbf{p}_k$ is a descent direction and $c_1 < 1$, it follows that $\psi'(0) = (1-c_1)\nabla f(\mathbf{x}_k)^T\mathbf{p}_k < 0$. Because ψ' is continuous and $f(\mathbf{x}_k + \alpha\mathbf{p}_k)$ is bounded from below there exists a smallest $\alpha^* > 0$ such that $\psi(\alpha^*) = 0$. It follows that $\psi(\alpha) \leq 0$ and hence (4.6) holds for all $\alpha \in]0, \alpha^*]$.

By the mean value theorem, there is $\alpha^{**} \in]0, \alpha^*[$ such that

$$f(\mathbf{x}_k + \alpha^*\mathbf{p}_k) - f(\mathbf{x}_k) = \alpha^*\nabla f(\mathbf{x}_k + \alpha^{**}\mathbf{p}_k)^T\mathbf{p}_k.$$

Combined with $\psi(\alpha^*) = 0$, this implies

$$\nabla f(\mathbf{x}_k + \alpha^{**}\mathbf{p}_k)^T\mathbf{p}_k = c_1\nabla f(\mathbf{x}_k)^T\mathbf{p}_k > c_2\nabla f(\mathbf{x}_k)^T\mathbf{p}_k.$$

Therefore, there is α^{**} satisfying the Wolfe conditions (4.6)–(4.7). By the continuous differentiability of f, they also hold for a (sufficiently small) interval around α^{**}. ∎

One of the great advantages of the Wolfe conditions is that they allow to prove convergence of the line search method (4.3) under fairly general assumptions.

Theorem 4.9 *Consider a line search method* (4.3)*, where $\mathbf{p}_k$ is a descent direction and α_k satisfies the the Wolfe conditions* (4.6)–(4.7) *in each iteration k. Suppose that f is bounded from below in $\mathbb{R}^n$ and continuously differentiable in an open set $\mathcal{U} \subset \mathbb{R}^n$ with $\{\mathbf{x} \mid f(\mathbf{x}) \leq f(\mathbf{x}_0)\} \subset \mathcal{U}$. Moreover, ∇f is assumed to be Lipschitz continuous on $\mathcal{U}$. Then*

$$\sum_{k=0}^{\infty} \cos^2\theta_k \cdot \|\nabla f(\mathbf{x}_k)\|_2^2 < \infty, \quad \textit{where} \quad \cos\theta_k := \frac{-\nabla f(\mathbf{x}_k)^T\mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|_2\|\mathbf{p}_k\|_2}. \tag{4.8}$$

Proof. The curvature condition (4.7) implies

$$\big(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\big)^T\mathbf{p}_k \geq (c_2 - 1)\nabla f(\mathbf{x}_k)^T\mathbf{p}_k,$$

On the other hand, the Lipschitz condition implies $\|\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\|_2 \leq L\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2$ and hence

$$\big(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\big)^T \mathbf{p}_k \leq \alpha_k L \|\mathbf{p}_k\|_2^2.$$

By combining both inequalities, we obtain

$$\alpha_k \geq \frac{c_2 - 1}{L} \frac{\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{\|\mathbf{p}_k\|_2^2}.$$

Using the Armijo condition (4.6) then yields

$$\begin{aligned} f(\mathbf{x}_{k+1}) &\leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k \\ &\leq f(\mathbf{x}_k) - c_1 \frac{1 - c_2}{L} \frac{\big(\nabla f(\mathbf{x}_k)^T \mathbf{p}_k\big)^2}{\|\mathbf{p}_k\|_2^2} \\ &= f(\mathbf{x}_k) - c \cos^2 \theta_k \cdot \|\nabla f(\mathbf{x}_k)\|_2^2, \end{aligned}$$

with $c = c_1 \frac{1-c_2}{L}$. Recursively inserting this relation gives

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_0) - c \sum_{j=0}^{k} \cos^2 \theta_k \cdot \|\nabla f(\mathbf{x}_k)\|_2^2.$$

Since $f(\mathbf{x}_0) - f(\mathbf{x}_{k+1})$ is bounded, the statement of the theorem follows by taking $k \to \infty$. □

Let us discuss the implications of Theorem 4.9. First of all, (4.8) implies

$$\cos^2 \theta_k \cdot \|\nabla f(\mathbf{x}_k)\|_2^2 \overset{k\to\infty}{\longrightarrow} 0. \tag{4.9}$$

(In fact, one can say a little more, e.g., the sequence must converge faster than $1/k$ to zero.) It is quite natural to assume that θ_k is bounded away from 90 degrees, that is, there is $\delta > 0$ such that

$$\cos \theta_k \geq \delta > 0, \quad \forall k.$$

For example, this is clearly the case for steepest descent, with $\delta = 1$. Under this condition, it immediately follows from (4.9) that

$$\|\nabla f(\mathbf{x}_k)\|_2^2 \overset{k\to\infty}{\longrightarrow} 0.$$

In other words, the line search method converges (globally) to a stationary point. Note that we cannot conclude that the method converges to a local minimizer. Making such a statement requires to inject additional information about the Hessian; this will lead to the Newton methods discussed in Section 4.2.4.

4.2.3 Rate of convergence for steepest descent

In the following, we aim at quantifying the (local) convergence speed for the steepest descent method. Let us first perform this analysis for a quadratic objective function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{4.10}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $\mathbf{b} \in \mathbb{R}^n$. This is about the best objective function one can dream up; it is strictly convex and $\mathbf{x}^* = A^{-1}\mathbf{b}$ is the global minimizer. Note that $\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$.

We consider an **exact line search** strategy, that is, α_k is chosen to minimize $f(\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k))$. This can be easily implemented for (4.10) as

$$\begin{aligned}\frac{\partial}{\partial\alpha} f(\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k)) &= -\nabla f(\mathbf{x}_k)^T A\mathbf{x}_k + \alpha\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k)^T\mathbf{b}\\ &= -\nabla f(\mathbf{x}_k)^T A\mathbf{x}_k + \alpha\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}_k)^T\mathbf{b}\\ &= -\nabla f(\mathbf{x}_k)^T\nabla f(\mathbf{x}_k) + \alpha\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k)\end{aligned}$$

is zero for

$$\alpha_k = \frac{\nabla f(\mathbf{x}_k)^T\nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k)}. \tag{4.11}$$

Hence, the steepest descent method for (4.10) with exact linear search takes the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\nabla f(\mathbf{x}_k)^T\nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k)}\nabla f(\mathbf{x}_k). \tag{4.12}$$

To quantify the convergence of (4.12), it will be convenient to measure the error in the norm induced by A: $\|\mathbf{y}\|_A^2 := \mathbf{y}^T A\mathbf{y}$.

Theorem 4.10 *The steepest descent method with exact line search applied to* (4.10) *satisfies*

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_A \le \frac{\kappa(A) - 1}{\kappa(A) + 1} \cdot \|\mathbf{x}_k - \mathbf{x}^*\|_A \tag{4.13}$$

where $\kappa(A) = \|A\|_2\|A\|_2^{-1} = \lambda_{\max}(A)/\lambda_{\min}(A)$ *denotes the condition number of* A.

Proof. Subtracting $\mathbf{x}^*$ on both sides of (4.11) gives

$$\mathbf{x}_{k+1} - \mathbf{x}^* = \mathbf{x}_k - \mathbf{x}^* - \frac{\nabla f(\mathbf{x}_k)^T\nabla f(\mathbf{x}_k)}{\nabla f(\mathbf{x}_k)^T A\nabla f(\mathbf{x}_k)}\nabla f(\mathbf{x}_k)$$

Letting $\mathbf{v} := \nabla f(\mathbf{x}_k) = A(\mathbf{x}_k - \mathbf{x}^*)$, we obtain

$$\rho := \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_A^2}{\|\mathbf{x}_k - \mathbf{x}^*\|_A^2} = \left(1 - \frac{\|\mathbf{v}\|_2^4}{\|\mathbf{v}\|_A^2\|\mathbf{x}_k - \mathbf{x}^*\|_A^2}\right)^2 = \left(1 - \frac{\|\mathbf{v}\|_2^4}{\|\mathbf{v}\|_A^2\|\mathbf{v}\|_{A^{-1}}^2}\right)^2 \tag{4.14}$$

To proceed further, we need the so called Kantorovich inequality,

$$\frac{\|\mathbf{v}\|_2^4}{\|\mathbf{v}\|_A^2\|\mathbf{v}\|_{A^{-1}}^2} \ge \frac{4\lambda_{\min}(A)\lambda_{\max}(A)}{\big(\lambda_{\min}(A) + \lambda_{\max}(A)\big)^2} = \frac{4\kappa(A)}{(1+\kappa(A))^2}, \tag{4.15}$$

which can be shown by noting that can restrict ourselves to vectors of the form $\mathbf{v} = \alpha\mathbf{v}_{\min} + \beta\mathbf{v}_{\max}$, where $\mathbf{v}_{\min}, \mathbf{v}_{\max}$ are eigenvectors belonging to $\lambda_{\min}(A), \lambda_{\max}(A)$. Combining (5.47) and (4.15) yields

$$\rho \leq \left(\frac{\kappa(A)-1}{\kappa(A)+1}\right)^2,$$

which concludes the proof. □

Let us now consider the case of general smooth f. By Taylor expansion of f around a strict local minimizer x^*, we have

$$f(\mathbf{x}_k) = f(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}_k - \mathbf{x}^*)^\mathsf{T} A(\mathbf{x}_k - \mathbf{x}^*) + O(\|\mathbf{x}_k - \mathbf{x}^*\|^3),$$

where $A = H(\mathbf{x}^*)$ is the Hessian at $\mathbf{x}^*$. Hence,

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) \approx \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}^*\|_A^2.$$

Moreover, one step of the steepest descent method will – in first order – produce nearly the same next iterate if we replace f by the quadratic model

$$f(\mathbf{x}) \approx f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x}_k - \mathbf{x}^*)^\mathsf{T} A(\mathbf{x}_k - \mathbf{x}^*), \tag{4.16}$$

provided that $\|\mathbf{x}_k - \mathbf{x}^*\|$ is sufficiently small. These considerations allow us to generalize Theorem 4.10.

Theorem 4.11 *Suppose that the steepest descent method with exact line search applied to a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ converges to a stationary point x^* with symmetric positive definite Hessian $H(\mathbf{x}^*)$. Then – for sufficiently large k – we have*

$$f(\mathbf{x}_{k+1}) \leq \rho^2\big(f(\mathbf{x}_k)\big),$$

for any

$$\rho = \frac{\kappa(H(\mathbf{x}^*))-1}{\kappa(H(\mathbf{x}^*))+1} + \epsilon < 1$$

with $\epsilon > 0$.

4.2.4 The Newton method

From the discussion in Section 4.2.2, one may be tempted to conclude that choosing $\cos\theta_k$ large is advisable and hence steepest descent will produce fastest convergence. Nothing could be more misleading! In this section, we discuss the (locally) much

faster Newton method for minimizing $f(\mathbf{x})$, which amounts to choosing the search direction

$$\mathbf{p}_k = -H(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k). \tag{4.17}$$

There are many different ways of motivating this choice for $\alpha_k = 1$. On the one hand, this amounts to the standard Newton method for solving the nonlinear equation $\nabla f(\mathbf{x}) = 0$. On the other hand, this minimizes the quadratic model (4.16) exactly. Both motivations indicate that the Newton method converges locally quadratically.

Theorem 4.12 *Consider a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ for which the Hessian is symmetric positive definite at a stationary point $\mathbf{x}^*$ and Lipschitz continuous in a neighborhood of $\mathbf{x}^*$. Then the following statements hold for the iteration $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ with the Newton direction* (4.17)*:*

1. *$\mathbf{x}_k \overset{k\to\infty}{\longrightarrow} \mathbf{x}^*$, provided that $\mathbf{x}_0$ is sufficiently close to x^*;*
2. *the sequence $\{\mathbf{x}_k\}$ converges locally quadratically;*
3. *the sequence $\{\|\nabla f(\mathbf{x}_k)\|\}$ converges locally quadratically to zero.*

Proof. By the definition of the Newton method, we have

$$\begin{aligned}\mathbf{x}_{k+1} - \mathbf{x}^* &= \mathbf{x}_k - \mathbf{x}^* + \mathbf{p}_k = \mathbf{x}_k - \mathbf{x}^* - H(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)\\ &= H(\mathbf{x}_k)^{-1}\left[H(\mathbf{x}_k)(\mathbf{x}_k - \mathbf{x}^*) - (\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}^*))\right].\end{aligned} \tag{4.18}$$

From the Taylor expansion and the Lipschitz continuity of H, we obtain a constant $L > 0$ such that

$$\|H(\mathbf{x}_k)(\mathbf{x}_k - \mathbf{x}^*) - (\nabla f(\mathbf{x}_k) - \nabla f(\mathbf{x}^*))\|_2 \le L\|\mathbf{x}_k - \mathbf{x}^*\|_2^2$$

holds for all $\mathbf{x}_k$ sufficiently close to $\mathbf{x}^*$. Combined with (4.18), this gives

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \le L\|H(\mathbf{x}_k)^{-1}\|_2\|\mathbf{x}_k - \mathbf{x}^*\|_2^2 \le \underbrace{2L\|H(\mathbf{x}^*)^{-1}\|_2}_{=:\tilde{L}}\|\mathbf{x}_k - \mathbf{x}^*\|_2^2, \tag{4.19}$$

where used the fact that $\|H(\mathbf{x}_k)^{-1}\|_2 \le 2\|H(\mathbf{x}^*)^{-1}\|_2$ for $\mathbf{x}_k$ sufficiently close to $\mathbf{x}^*$. The inequality (4.19) shows the local quadratic convergence of $\mathbf{x}_k$.

It remains to prove the local quadratic convergence of $\{\|\nabla f(\mathbf{x}_k)\|\}$. This is shown by the same arguments as above:

$$\begin{aligned}\|\nabla f(\mathbf{x}_{k+1})\|_2 &= \|\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) - H(\mathbf{x}_k)\mathbf{p}_k\|_2\\ &\le L\|\mathbf{p}_k\|^2 \le L\|H(\mathbf{x}_k)^{-1}\|_2^2\|\nabla f(\mathbf{x}_k)\|_2^2\\ &\le 2L\|H(\mathbf{x}^*)^{-1}\|_2^2\|\nabla f(\mathbf{x}_k)\|_2^2\end{aligned}$$

where we used $\nabla f(\mathbf{x}_k) + H(\mathbf{x}_k)\mathbf{p}_k = 0$. ∎

As shown by Theorem 4.12, choosing the step length $\alpha_k = 1$ yields *local* quadratic convergence. In general, $\alpha_k = 1$ is not a good choice in the beginning of the iteration. In practice, the Newton method should therefore be combined with the Armijo or the Wolfe conditions. The expectation is that, initially, α_k is less than 1. Once the region of local quadratic convergence for the Newton method is reached, the conditions allow for choosing $\alpha_k = 1$. This indicates that local quadratic convergence will also be attained in such a setting, see [UU] for the precise statement.

4.2.5 Quasi-Newton methods

The computation of the Newton direction $\mathbf{p}_k = -H(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$ is often too expensive, due to the need for determining the Hessian and solving a linear system. The general idea of **quasi-Newton methods** is to approximate $H(\mathbf{x}_k)$ by a symmetric positive matrix B_k, leading to a search direction of the form

$$\mathbf{p}_k = -B_k^{-1}\nabla f(\mathbf{x}_k). \tag{4.20}$$

It is important to quantify the extent to which B_k shall approximate $H(\mathbf{x}_k)$ to obtain good convergence, that is, faster convergence than the steepest descent method. As we will see below, it is sufficient to require that B_k provides an increasingly accurate approximation of $H(\mathbf{x}_k)$ *along the search direction* $\mathbf{p}_k$:

$$\lim_{k\to\infty} \frac{\|(B_k - H(\mathbf{x}_k))\mathbf{p}_k\|_2}{\|\mathbf{p}_k\|_2} = 0. \tag{4.21}$$

Theorem 4.13 *Consider a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ for which the Hessian is symmetric positive definite at a stationary point $\mathbf{x}^*$ and Lipschitz continuous in a neighborhood of $\mathbf{x}^*$. Suppose that the iteration $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ with the quasi-Newton direction (4.20) converges to $\mathbf{x}^*$. Then $\{\mathbf{x}_k\}$ converges superlinearly if and only if (4.21) holds.*

Proof. The key idea of the proof is to relate the quasi-Newton direction to the Newton direction $\mathbf{p}_k^N := -H(\mathbf{x}_k)^{-1}\nabla f(\mathbf{x}_k)$. Assuming that (4.21) holds, we have

$$\begin{aligned}\|\mathbf{p}_k - \mathbf{p}_k^N\|_2 &= \big\|H(\mathbf{x}_k)^{-1}\big(H(\mathbf{x}_k)\mathbf{p}_k + \nabla f(\mathbf{x}_k)\big)\big\|_2 \\ &\le \|H(\mathbf{x}_k)^{-1}\|_2\|(H(\mathbf{x}_k) - B_k)\mathbf{p}_k\|_2 = o(\|\mathbf{p}_k\|_2).\end{aligned}$$

On the other hand, $\|\mathbf{p}_k - \mathbf{p}_k^N\|_2 = o(\|\mathbf{p}_k\|_2)$ immediately implies (4.21). Hence, both conditions are equivalent.

By using the result of Theorem 4.12, we thus obtain

$$\begin{aligned}\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2 = \|\mathbf{x}_k + \mathbf{p}_k - \mathbf{x}^*\|_2 &\le \|\mathbf{x}_k + \mathbf{p}_k^N - \mathbf{x}^*\|_2 + \|\mathbf{p}_k - \mathbf{p}_k^N\|_2 \\ &\le O(\|\mathbf{x}_k - \mathbf{x}^*\|^2) + o(\|\mathbf{p}_k\|_2).\end{aligned}$$

This proves superlinear convergence if (4.21) holds. The other direction of the statement is an immediate consequence of the fact that superlinear convergence is only possible if $\|\mathbf{p}_k - \mathbf{p}_k^N\|_2 = o(\|\mathbf{p}_k\|_2)$. □

There is a lot of freedom in choosing B_k. Quasi-Newton methods choose a sequence $B_0, B_1, B_2, \ldots$ satisfying the condition

$$B_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k) \tag{4.22}$$

starting from an inital symmetric positive definite matrix B_0 (which preferably is an approximation of $H(\mathbf{x}_0)$). Note that (4.22) mimicks the approximation of a tangent vector by the secant vector.

Even when imposing (4.22), there remains a lot of freedom. One usually restricts the freedom further by requiring the update $B_{k+1} - B_k$ to be a low-rank matrix, which allows for the efficient inversion of B_{k+1} using the inverse of B_k. When requiring the update to be symmetric and of rank 1, the choice of B_{k+1} becomes unique:

$$B_{k+1} = B_k + \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)(\mathbf{y}_k - B_k\mathbf{s}_k)^T}{(\mathbf{y}_k - B_k\mathbf{s}_k)^T\mathbf{s}_k}, \tag{4.23}$$

where

$$\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k = \alpha_k\mathbf{p}_k, \quad \mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k).$$

The quasi-Newton method resulting from (4.23) is called **SR1** (symmetric rank-1). Some care needs to be applied when using SR1; the denominator $(\mathbf{y}_k - B_k\mathbf{s}_k)^T\mathbf{s}_k$ may become negative (or even zero!), potentially destroying positive definitness.

By far, the most popular quasi-Newton method is **BFGS** (Broyden-Fletcher-Goldfarb-Shanno):

$$B_{k+1} = B_k + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} - \frac{(B_k\mathbf{s}_k)(B_k\mathbf{s}_k)^T}{\mathbf{s}_k^T B_k\mathbf{s}_k}. \tag{4.24}$$

It can be easily seen that this update satisfies (4.22). Much can (and should) be said about the properties of BFGS. However, the analysis of BFGS is significantly more complicated than the analysis of the Newton method, due to the evolution of B_k. Under suitable conditions, it can be shown that BFGS satisfies (4.21) and hence converges superlinearly.

4.3 The nonlinear conjugate gradient method

4.3.1 The linear conjugate gradient method

Let us recall the (linear) conjugate gradient method for the objective function

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

with a symmetric positive definite matrix A. Recall that the gradient of f at $\mathbf{x}_k$ is given by

$$\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b}_k.$$

In Section 4.2.3 we have already seen and analysed the method of steepest descent for this problem. We also seen that it exhibits a 'zigzag' behavior for ill-conditioned matrices, resulting in slow convergence. This 'zigzag' behavior can be avoided by choosing the search directions $\{\mathbf{p}_0, \mathbf{p}_1, \ldots\}$ orthogonal to each other, in the inner product induced by A:

$$\mathbf{p}_i^T A \mathbf{p}_j = 0 \qquad \forall i \neq j. \tag{4.25}$$

Then we generate a sequence $\{\mathbf{x}_k\}$ by setting

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

with parameter α_k obtained from exact line search:

$$\alpha_k = -\frac{\mathbf{r}_k^T \mathbf{p}_k}{\mathbf{p}_k^T A \mathbf{p}_k}.$$

Because of (4.25), this is sometimes called **conjugate directions method**.

In the **conjugate gradient method**, the directions are chosen such that

$$\operatorname{span}\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\} = \operatorname{span}\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_k\} \tag{4.26}$$

holds, that is, the search directions span the same space as the gradients. To generate such directions, let us suppose we can do this via a recursion of the form

$$\mathbf{p}_k = -\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}.$$

The condition $\mathbf{p}_{k-1}^T A \mathbf{p}_k = 0$ implies

$$\beta_k = \frac{\mathbf{r}_k^T A \mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^T A \mathbf{p}_{k-1}}$$

It then follows that (4.25) and (4.26) hold (which is by no means trivial to show). Moreover, it can be shown that

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T A \mathbf{p}_k}, \qquad \beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k},$$

which avoids unnecessary multiplications with A in the computation of these scalars. Putting everything together yields Algorithm 4.14.

Algorithm 4.14 CG method

Input: Symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$. Starting vector $\mathbf{x}_0 \in \mathbb{R}^n$. $k_{\max} \in \mathbb{N}$.

Output: Approximate solution $\mathbf{x}_k$ to $A\mathbf{x} = \mathbf{b}$.

$\mathbf{r}_0 \leftarrow \mathbf{b} - A\mathbf{x}_0$, $\mathbf{p}_0 \leftarrow -\mathbf{r}_0$
for $k = 0, 1, \ldots, k_{\max}$ **do**

$\alpha_k \leftarrow \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T A \mathbf{p}_k}$
$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$
$\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k + \alpha_k A \mathbf{p}_k$
$\beta_{k+1} \leftarrow \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
$\mathbf{p}_{k+1} \leftarrow -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k$
end for

It is informative to compare the following convergence result with Theorem 4.13.

Theorem 4.15 *Let $\mathbf{x}_k$ denote the approximate solution obtained after applying k steps of CG with starting vector $\mathbf{x}_0$. Then*

$$\frac{\|\mathbf{x} - \mathbf{x}_k\|_A}{\|\mathbf{x} - \mathbf{x}_0\|_A} \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k.$$

4.3.2 The Fletcher-Reeves method

Algorithm 4.14 can be applied to a general nonlinear optimization problem by simply replacing $\mathbf{r}_k$ with a gradient. Only one additional change is necessary, as it is in general not possible to obtain the step length α_k by exact line search. This can be replaced by, e.g., the Armijo rule. As a result, we obtain Algorithm 4.16.

Algorithm 4.16 Fletcher-Reeves method
Input: Objective function f. Starting vector $\mathbf{x}_0 \in \mathbb{R}^n$. $k_{\max} \in \mathbb{N}$.
Output: Approximate minimizer $\mathbf{x}_k$ of f.

Evaluate $\nabla_0 = \nabla f(\mathbf{x}_0)$ and set $\mathbf{p}_0 \leftarrow -\nabla_0$
for $k = 0, 1, \ldots, k_{\max}$ **do**
Compute α_k by line search and set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$
Evaluate $\nabla_{k+1} = \nabla f(\mathbf{x}_{k+1})$
$\beta_{k+1}^{\mathrm{FR}} \leftarrow \frac{\nabla_{k+1}^T \nabla_{k+1}}{\nabla_k^T \nabla_k}$
$\mathbf{p}_{k+1} \leftarrow -\nabla_{k+1} + \beta_{k+1}^{\mathrm{FR}} \mathbf{p}_k$
end for

It would be comforting to know that the directions produced by Algorithm 4.16 are indeed descent directions. To check this, we compute

$$\nabla f(\mathbf{x}_k)^T \mathbf{p}_k = -\|\nabla f(\mathbf{x}_k)\|^2 + \beta_k^{\mathrm{FR}} \nabla f(\mathbf{x}_k)^T \mathbf{p}_{k-1}. \tag{4.27}$$

If the line search is exact, we have $\nabla f(\mathbf{x}_k)^T \mathbf{p}_{k-1} = 0$ and hence (4.27) is always negative. For inexact line search this is not clear at all. Fortunately, it will turn

out to be the case if we impose the **strong Wolfe conditions**:

$$f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) - f(\mathbf{x}_k) \leq c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{p}_k, \tag{4.28}$$

$$|\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)^T \mathbf{p}_k| \leq -c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k.. \tag{4.29}$$

While (4.6) and (4.28) are identical, the absolute value in (4.29) is not present in (4.7). Moreover, we impose $0 < c_1 < c_2 < \frac{1}{2}$ (instead of $0 < c_1 < c_2 < 1$).

Theorem 4.17 *Suppose that Algorithm 4.16 makes use of a step length α_k satisfying* (4.29) *with $c_2 < 1/2$. Then the method generates descent directions that satisfy*

$$-\frac{1}{1-c_2} \leq \frac{\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|_2^2} \leq \frac{2c_2 - 1}{1 - c_2}. \tag{4.30}$$

Proof. By elementary considerations,

$$-1 < \frac{2c_2 - 1}{1 - c_2} < 0, \tag{4.31}$$

and we therefore obtain the descent property once (4.30) is established.

The proof of (4.30) is by induction in k. The case $k = 0$ follows from (4.31). Assume now that (4.30) holds for some k. Then, by Algorithm 4.16,

$$\frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_{k+1}}{\|\nabla f(\mathbf{x}_{k+1})\|_2^2} = -1 + \beta_{k+1}^{\mathrm{FR}} \frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_{k+1})\|_2^2} = -1 + \beta_{k+1}^{\mathrm{FR}} \frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|_2^2}.$$

Combining this with the curvature condition (4.29),

$$|\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_k| \leq -c_2 \nabla f(\mathbf{x}_k)^T \mathbf{p}_k,$$

we obtain

$$-1 + c_2 \frac{\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|_2^2} \leq \frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_{k+1}}{\|\nabla f(\mathbf{x}_{k+1})\|_2^2} \leq -1 - c_2 \frac{\nabla f(\mathbf{x}_k)^T \mathbf{p}_k}{\|\nabla f(\mathbf{x}_k)\|_2^2}.$$

Using the induction hypothesis yields

$$-1 - \frac{c_2}{1 - c_2} \leq \frac{\nabla f(\mathbf{x}_{k+1})^T \mathbf{p}_{k+1}}{\|\nabla f(\mathbf{x}_{k+1})\|_2^2} \leq -1 + \frac{c_2}{1 - c_2},$$

which completes the proof. □

Note that Theorem 4.17 does not make use of the Armijo condition (4.28). This condition is still needed, to ensure global convergence. However, in contrast to 4.15, the global convergence statements for nonlinear CG methods are much weaker; see Chapter 5 in [NW].

4.3.3 The Polak-Ribière method

If it happens that $\mathbf{p}_k$ is a very poor search direction, that is, it is nearly orthogonal to $\mathbf{p}_k$ then Algorithm 4.16 makes very little progress in one step and hence $\mathbf{x}_{k+1} \approx \mathbf{x}_k$, $\nabla_{k+1} \approx \nabla_k$. So, we have

$$\beta_{k+1}^{\mathrm{FR}} = \frac{\nabla_{k+1}^T \nabla_{k+1}}{\nabla_k^T \nabla_k} \approx 1.$$

Moreover, it can be shown that, in such a situation, $\|\nabla_k\|$ (and therefore also $\|\nabla_{k+1}\|$) needs to be tiny for (4.29) to be satisfied. Consequently,

$$\mathbf{p}_{k+1} \approx \mathbf{p}_k$$

and Algorithm 4.16 will also make very little progress in the next step. In other words, it gets stuck.

The **Polak-Ribière method** aims to avoid the described situation by replacing $\beta_{k+1}^{\mathrm{FR}}$ with

$$\beta_{k+1}^{\mathrm{PR}} = \frac{\nabla_{k+1}^T (\nabla_{k+1} - \nabla_k)}{\nabla_k^T \nabla_k}.$$

The Fletcher-Reeves and Polak-Ribière methods with *exact* line search are identical for strongly convex functions. In all other situations, they can differ, sometimes to a large extent. Often, the Polak-Ribière method yields more robust and faster convergence.

There are several other strategies for choosing β_{k+1}, see [NW].

4.4 Smooth convex optimization

All convergence results presented so far have a local nature and do not guarantee *global* convergence to a *global* minimum. More can be said if the target functional is convex. This section mainly follows Chapter 2 from [N] but it also includes some of the discussion from `https://blogs.princeton.edu/imabandit/orf523-the-complexities-of-optimization/`.

4.4.1 Convex functions

Definition 4.18 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is called* convex *if*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}) \tag{4.32}$$

holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

It is often quite tedious to verify the condition (4.32). It is much easier to check convexity for (twice) continuously differentiable functions.

Lemma 4.19 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. f is convex if and only if one of the following three conditions holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:*

1.
$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}), \tag{4.33}$$

2.
$$(\nabla f(\mathbf{y}) - \nabla f(\mathbf{x}))^T(\mathbf{y}-\mathbf{x}) \geq 0, \tag{4.34}$$

3.
$$H(\mathbf{x}) \geq 0, \tag{4.35}$$

where we additionally assume for (4.35) that f is twice continuously differentiable.

Proof. In the following, we set $\mathbf{x}_\alpha = \alpha\mathbf{x} + (1-\alpha)\mathbf{y}$.

1. (4.32) $\Rightarrow$ (4.33): Rearranging (4.32) gives for any $\alpha \in [0,1[$:

$$f(\mathbf{y}) \geq \frac{1}{1-\alpha}\big(f(\mathbf{x}_\alpha) - \alpha f(\mathbf{x})\big) = f(\mathbf{x}) + \frac{1}{1-\alpha}\big(f(\mathbf{x} + (1-\alpha)(\mathbf{y}-\mathbf{x})) - f(\mathbf{x})\big).$$

Letting $\alpha \to 1$ implies (4.33).

(4.33) $\Rightarrow$ (4.32). Applying (4.33) to $\mathbf{y}, \mathbf{x}_\alpha$ and $\mathbf{x}, \mathbf{x}_\alpha$, respectively, gives

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}_\alpha) + \nabla f(\mathbf{x}_\alpha)^T(\mathbf{y}-\mathbf{x}_\alpha), \\ f(\mathbf{x}) &\geq f(\mathbf{x}_\alpha) + \nabla f(\mathbf{x}_\alpha)^T(\mathbf{x}-\mathbf{x}_\alpha). \end{aligned}$$

Multiplying these inequalities with α and $1-\alpha$, respectively, and adding the results up yields (4.32).

2. (4.33) $\Rightarrow$ (4.34): Reversing the roles of $\mathbf{x}$ and $\mathbf{y}$ in (4.33) gives the ineqality $-f(\mathbf{y}) \geq -f(\mathbf{x}) - \nabla f(\mathbf{y})^T(\mathbf{y}-\mathbf{x})$, which – when added to (4.33) – yields (4.34).

(4.34) ⇒ (4.33): This implication follows from

$$\begin{aligned} f(\mathbf{y}) - f(\mathbf{x}) &= \int_0^1 \nabla f(\mathbf{x}_\alpha)^T(\mathbf{y}-\mathbf{x})\,\mathrm{d}\alpha \\ &= \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \int_0^1 \big(\nabla f(\mathbf{x}_\alpha) - \nabla f(\mathbf{x})\big)^T(\mathbf{y}-\mathbf{x})\,\mathrm{d}\alpha \\ &= \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{1}{1-\alpha}\int_0^1 \big(\nabla f(\mathbf{x}_\alpha) - \nabla f(\mathbf{x})\big)^T(\mathbf{x}_\alpha-\mathbf{x})\,\mathrm{d}\alpha \\ &\geq \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}). \end{aligned}$$

3. (4.34) ⇒ (4.35): Let $\mathbf{x}_\tau = \mathbf{x} + \tau\mathbf{s}$ for an arbitrary vector $\mathbf{s} \in \mathbb{R}^n$ and $\tau > 0$. Then (4.34) yields

$$0 \leq \frac{1}{\tau^2}\big(\nabla f(\mathbf{x}_\tau) - \nabla f(\mathbf{x})\big)^T(\mathbf{x}_\tau - \mathbf{x}) = \frac{1}{\tau}\big(\nabla f(\mathbf{x}_\tau) - \nabla f(\mathbf{x})\big)^T\mathbf{s} \overset{\tau\to 0}{\longrightarrow} \mathbf{s}^T H(\mathbf{x})\mathbf{s},$$

which implies positive semidefiniteness of $H(\mathbf{x})$.

4. (4.35) ⇒ (4.34): This implication follows from the Taylor expansion, with an integral representation of the remainder term. □

Examples of convex functions:

- The univariate functions e^x, $|x|^p$ for $p \geq 1$, $\frac{x^2}{1-|x|}$, $|x| - \log(1+|x|)$ are convex.
- The norm properties imply that *any* vector norm on $\mathbb{R}^n$ is convex. In particular, this holds for $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_1$.
- The function $f(\mathbf{x}) = \alpha + \mathbf{b}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T A\mathbf{x}$ is convex if A is symmetric positive semidefinite. In particular, any linear function $\alpha + \mathbf{b}^T\mathbf{x}$ is convex.
- If f_1, f_2 are convex functions then the functions $f_1(\mathbf{x}) + f_2(\mathbf{x})$ and $\max\{f_1(\mathbf{x}), f_2(\mathbf{x})\}$ are convex as well.
- If f is convex then the function $\varphi(\mathbf{x}) := f(A\mathbf{x} + \mathbf{b})$ is convex as well for any matrix A and vector $\mathbf{b}$ of suitable size.

The following result is one of the main reasons for the importance of convex functions.

Theorem 4.20 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and continuously differentiable. Then x^* is a global minimizer for f if and only if $\nabla f(\mathbf{x}^*) = 0$.*

Proof. One direction follows immediately from Theorem 4.1. For the other direction, suppose that $\nabla f(\mathbf{x}^*) = 0$. Then (4.33) immediately implies $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for any $\mathbf{x} \in \mathbb{R}^n$ and thus $\mathbf{x}^*$ is a global minimizer. □

4.4.2 Strongly convex functions

We already know that the local convergence rate of gradient descent methods depends on the condition number of $H(\mathbf{x}^*)$. Since convexity does not necessarily imply that this condition number is finite; we need to introduce a stronger concept to ensure fast convergence of gradient descent.

Definition 4.21 *A continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is called* strongly convex *if there is $\mu > 0$ such that*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(x)^T(\mathbf{y} - \mathbf{x}) + \frac{1}{2}\mu\|\mathbf{y} - \mathbf{x}\|_2^2$$

holds for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Later on, the value of μ will play an important role and we will sometimes say *μ-strongly convex function*. When adding a μ_1-strongly convex function with a μ_2-strongly convex function, one obtains a $(\mu_1 + \mu_2)$-strongly convex function.

An immediate consequence of Definition 4.21, we have

$$f(\mathbf{x}) \geq f(\mathbf{x}^*) + \frac{1}{2}\mu\|\mathbf{x} - \mathbf{x}^*\|_2^2$$

at a minimizer $\mathbf{x}^*$. Thus, the minimizer $\mathbf{x}^*$ is uniquely determined.

The following lemma extends Lemma 4.19 and can be proven in a similar manner.

Lemma 4.22 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable. f is μ-strongly convex if and only if one of the following three conditions holds for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$:*

1. $f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha)f(\mathbf{y}) - \alpha(1-\alpha)\frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 \quad \forall \alpha \in [0,1]$,
2. $(\nabla f(\mathbf{y}) - \nabla f(\mathbf{x}))^T(\mathbf{y} - \mathbf{x}) \geq \mu\|\mathbf{x} - \mathbf{y}\|_2^2$,
3. $H(\mathbf{x}) \geq \mu I_n$,

where we additionally assume for the last case that f is twice continuously differentiable.

4.4.3 Nesterov's accelerated gradient descent method

The purpose of the line search strategies from Section 4.2 was to ensure global convergence to a stationary point. This is much less of a concern for convex functions, for which a constant step size is sufficient.

The following theorem summarizes the results from Section 2.1.5 of [N]. Note that we call a function *L-smooth* if it is continously differentiable and its gradient is Lipschitz continuous with Lipschitz constant L:

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

If f is twice continuously differentiable, this is equivalent to $\|H(\mathbf{x})\|_2 \leq L$ for all $\mathbf{x} \in \mathbb{R}^n$.

Theorem 4.23 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function with minimum $f^* = f(\mathbf{x}^*)$. Then the sequence generated by $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)$ for some initial vector $\mathbf{x}_0 \in \mathbb{R}^n$ has the following properties:*

1. *if $\alpha = \frac{1}{L}$ then $f(\mathbf{x}_k) - f^* \leq \frac{2L\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{k+4}$;*
2. *if f is μ-strongly convex and $\alpha = \frac{2}{\mu + L}$ then*

$$\|\mathbf{x}_k - \mathbf{x}^*\|_2^2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2,$$

$$f(\mathbf{x}_k) - f^* \leq \frac{L}{2}\left(\frac{\kappa - 1}{\kappa + 1}\right)^{2k} \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2,$$

where $\kappa = L/\mu$.

The convergence predicted in the first part of the theorem is algebraic of order 1: $O(1/k)$. This is not optimal and can be improved with an adapted choice of α, as we will see below. The convergence predicted in the second part of the theorem is very similar to Theorem 4.9, with a difference of major importance: Theorem 4.9 is a statement about local convergence, while the second part of Theorem 4.23 ensures global (exponential) convergence. It turns out that the convergence rates predicted by Theorem 4.23 can be improved significantly by adjusting the step size.

The idea of the accelerated gradient method, Algorithm 4.24, is to not fully accept the current iterate proposed by steepest descent (with step size $1/L$) but combine it with the old iterate. The parameter γ_k determining this convex combination arises from optimality considerations that can be found in [N].

Algorithm 4.24 Accelerated gradient descent for convex functions I
Input: *L-smooth convex function f and starting vector $\mathbf{x}_0$.*
Output: *Vector $\mathbf{y}_k$ approximating minimum: $f(\mathbf{y}_k) \approx f(\mathbf{x}^*)$.*

Set $\lambda_0 = 1$ *and* $\mathbf{y}_0 = \mathbf{x}_0$.
for $k = 0, 1, 2, \ldots$ ***do***
  Set $\lambda_{k+1} = \frac{1+\sqrt{1+4\lambda_k^2}}{2}$.
  Set $\gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}}$.
  Set $\mathbf{y}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$.
  Set $\mathbf{x}_{k+1} = (1-\gamma_k)\mathbf{y}_{k+1} + \gamma_k \mathbf{y}_k$.
end for

For analysing the convergence of Algorithm 4.24, we need the following important result on smooth convex functions.

Lemma 4.25 *For an L-smooth convex function f, we have*

$$f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) \leq \frac{L}{2}\|\mathbf{y}-\mathbf{x}\|_2^2$$

for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

Proof. By Taylor expansion with an integral representation of the remainder term, we obtain

$$f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(x)^T(\mathbf{y}-\mathbf{x}) + \int_0^1 \big(\nabla f(\mathbf{x}+\tau(\mathbf{y}-\mathbf{x}) - \nabla f(\mathbf{x})\big)^T(\mathbf{y}-\mathbf{x})\ \mathrm{d}\tau.$$

By convexity, $f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(x)^T(\mathbf{y}-\mathbf{x}) \geq 0$ and, using the Cauchy-Schwarz inequality, we get

$$\begin{aligned} &f(\mathbf{y}) - f(\mathbf{x}) - \nabla f(x)^T(\mathbf{y}-\mathbf{x}) \\ &\leq \int_0^1 \big\|\nabla f(\mathbf{x}+\tau(\mathbf{y}-\mathbf{x}) - \nabla f(\mathbf{x})\big\|_2 \|\mathbf{y}-\mathbf{x}\|_2\ \mathrm{d}\tau \leq \frac{L}{2}\|\mathbf{y}-\mathbf{x}\|_2^2. \end{aligned}$$

□

Theorem 4.26 *For an L-smooth convex function f, the iterates produced by Algorithm 4.24 satisfy*

$$f(\mathbf{y}_{k+1}) - f(\mathbf{x}^*) \leq \frac{2L\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{(k+2)^2}.$$

Proof. By Lemma 4.25,

$$\begin{aligned} f(\mathbf{y}_{k+1}) - f(\mathbf{x}_k) &= f\Big(\mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)\Big) - f(\mathbf{x}_k) \\ &\le \nabla f(\mathbf{x}_k)^T\Big(-\frac{1}{L}\nabla f(\mathbf{x}_k)\Big) + \frac{L}{2}\Big\|\frac{1}{L}\nabla f(\mathbf{x}_k)\Big\|_2^2 \\ &\le -\frac{1}{2L}\|\nabla f(\mathbf{x}_k)\|_2^2. \end{aligned}$$

This implies, using Lemma 4.19.1,

$$\begin{aligned} f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k) &\le f(\mathbf{y}_{k+1}) - f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{y}_k) \\ &\le -\frac{1}{2L}\|\nabla f(\mathbf{x}_k)\|_2^2 + \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{y}_k) \end{aligned}$$

and, similarly,

$$f(\mathbf{y}_{k+1}) - f(\mathbf{x}^*) \le -\frac{1}{2L}\|\nabla f(\mathbf{x}_k)\|_2^2 + \nabla f(\mathbf{x}_k)^T(\mathbf{x}_k - \mathbf{x}^*).$$

Setting $\delta_k = f(\mathbf{y}_k) - f(\mathbf{x}^*)$, a convex combination of these two inequalities gives

$$\begin{aligned} \lambda_k\delta_{k+1} - (\lambda_k - 1)\delta_k &= (\lambda_k - 1)(f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k)) + f(\mathbf{y}_{k+1}) - f(\mathbf{x}^*) \\ &\le -\frac{\lambda_k}{2L}\|\nabla f(\mathbf{x}_k)\|_2^2 + \nabla f(\mathbf{x}_k)^T\big(\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*\big). \end{aligned}$$

By definition, λ_k is defined from λ_{k-1} by satisfying the quadratic equation $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$. Multiplying the last inequality with λ_k, this gives

$$\begin{aligned} &\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \\ &\le -\frac{\lambda_k^2}{2L}\|\nabla f(\mathbf{x}_k)\|_2^2 + \lambda_k\nabla f(\mathbf{x}_k)^T\big(\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*\big) \\ &= -\frac{L}{2}\Big(\|\lambda_k(\mathbf{y}_{k+1} - \mathbf{x}_k)\|_2^2 + 2\lambda_k(\mathbf{y}_{k+1} - \mathbf{x}_k)^T(\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*)\Big). \end{aligned}$$

By basic manipulation, one can see that

$$\begin{aligned} &\|\lambda_k(\mathbf{y}_{k+1} - \mathbf{x}_k)\|_2^2 + 2\lambda_k(\mathbf{y}_{k+1} - \mathbf{x}_k)^T(\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^* \\ &= \|\lambda_k\mathbf{y}_{k+1} - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*\|_2^2 - \|\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*\|_2^2 \\ &= \|\lambda_{k+1}\mathbf{x}_{k+1} - (\lambda_{k+1} - 1)\mathbf{y}_{k+1} - \mathbf{x}^*\|_2^2 - \|\lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*\|_2^2 \\ &= \|\mathbf{u}_{k+1}\|_2^2 - \|\mathbf{u}_k\|_2^2, \end{aligned}$$

where we set $\mathbf{u}_k := \lambda_k\mathbf{x}_k - (\lambda_k - 1)\mathbf{y}_k - \mathbf{x}^*$. In summary, we have

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \le \frac{L}{2}\big(\|\mathbf{u}_k\|_2^2 - \|\mathbf{u}_{k+1}\|_2^2\big).$$

Formally setting $\lambda_{-1} = 0$ and summing up this inequality for $0, \dots, k$ yields

$$\lambda_k^2\delta_{k+1} \le \frac{L}{2}\big(\|\mathbf{u}_0\|_2^2 - \|\mathbf{u}_{k+1}\|_2^2\big) \le \frac{L}{2}\|\mathbf{u}_0\|_2^2 = \frac{L}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2.$$

We will now show by induction that $\lambda_k \geq (k+2)/2$. The ineqality is obviously satisfied for $k = 0$. The induction step follows from

$$\lambda_{k+1} = \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2} \geq \frac{1 + \sqrt{1 + (k+2)^2}}{2} \geq \frac{k+3}{2}.$$

This concludes the proof. □

Remark 4.27 When implementing Algorithm 4.24, it is important to supply a (reasonably tight) upper bound for the Lipschitz constant L. In specific cases, such a bound may be available, but in general it needs to be estimated. We refer to [P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. 2008] for a backtracking procedure for estimating L. ◇

Theorem 4.26 constitutes a significant improvement compared to the expected convergence of standard gradient descent applied to a smooth convex function, see Part 1 of Theorem 4.20. It would be quite satisfactory if we also achieved an improvement for μ-*strongly* convex functions compared to Part 2 of Theorem 4.20. It turns out that Algorithm 4.24 does *not* achieve this goal. For this purpose, one needs to incoporate (a reasonably tight lower bound of) μ. We refer to [B. O'Donoghue and E. Candès. Adaptive Restart for Accelerated Gradient Schemes. 2012] for a more detailed discussion, which also discusses the estimation of μ. Algorithm 4.28 summarizes the resulting procedure.

Algorithm 4.28 Accelerated gradient descent for convex functions II
Input: *L-smooth μ-strongly convex function f and starting vector $\mathbf{x}_0$.*
Output: *Vector $\mathbf{y}_k$ approximating minimum: $f(\mathbf{y}_k) \approx f(\mathbf{x}^*)$.*

Set $\mathbf{y}_0 = \mathbf{x}_0$.
for $k = 0, 1, 2, \ldots$ ***do***
  Set $\mathbf{y}_{k+1} = \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k)$.
  Set $\mathbf{x}_{k+1} = \mathbf{y}_{k+1} + \frac{1-\sqrt{\mu/L}}{1+\sqrt{\mu/L}}(\mathbf{y}_{k+1} - \mathbf{y}_k)$.
end for

Theorem 4.29 *For an L-smooth μ-strongly convex function f, the iterates produced by Algorithm 4.24 satisfy*

$$f(\mathbf{y}_k) - f(\mathbf{x}^*) \leq \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left(f(\mathbf{x}_0) - f^* + \frac{\mu}{2}\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2\right)$$
$$\leq L\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2.$$

Proof. The first inequality is Theorem 2.2.3 in [N]. The second inequality follows from the first using Lemma 4.25. □

Chapter 5

Constrained optimization

A **constrained optimization problem** takes the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \text{subject to} \quad g(\mathbf{x}) \geq 0,\ h(\mathbf{x}) = 0, \tag{5.1}$$

with $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$.

The conditions $g(\mathbf{x}) \geq 0$ and $h(\mathbf{x}) = 0$ impose constraints on $\mathbf{x}$, which are called **inequality constraints** and **equality constraints**, respectively. By defining the **feasible set**

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \geq 0,\ h(\mathbf{x}) = 0\} \tag{5.2}$$

we can rewrite (5.1) as

$$\min_{\mathbf{x}\in\Omega} f(\mathbf{x}),$$

which makes it visually closer to an unconstrained optimization problem.

5.1 Fundamentals

The goal of this section is to develop some theoretical understanding of the constrained optimization problem (5.1).

There may be the naive hope that imposing constraints makes it easier to solve an optimization problem, for example, because the constraints exclude local minima that are not global minima. Nearly always, the opposite is the case: imposing constraints makes optimization much harder! For example[3], consider the problem

$$\min_{\mathbf{x}\in\mathbb{R}^2} (x_2 + 100)^2 + 0.01x_1^2 \quad \text{subject to} \quad x_2 - \cos x_1 \geq 0. \tag{5.3}$$

Without the constraint, the (unique) local minimum is given by $(0, -100)^T$. With the constraints, there are local solutions near the points $\mathbf{x}^{(k)} = (k\pi, -1)^T$ for all odd integers k, see Figure 5.1.

[3]This and the following examples are taken from [NW].

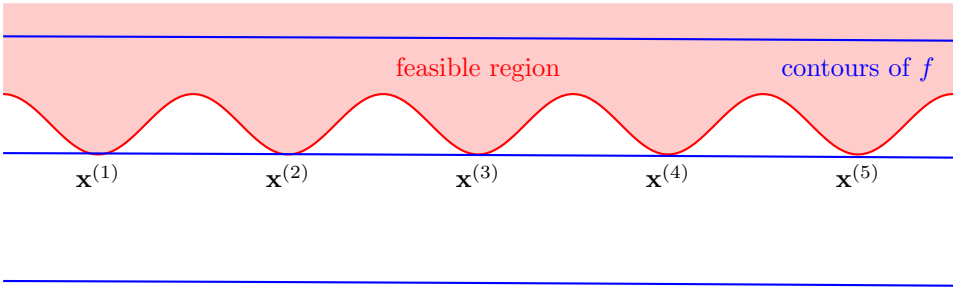


Figure 5.1. *Local solutions of* (5.3).

Note that the concept of local solutions corresponds to the notion of local minima from Chapter 4 restricted to the feasible set Ω defined in (5.2). A point $\mathbf{x}^* \in \Omega$ is called a **local solution** of (5.1) if there is a neighborhood $\mathcal{N}$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N} \cap \Omega$. Similarly, $\mathbf{x}^*$ is called a **strict local solution** if $f(\mathbf{x}^*) < f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N} \cap \Omega$ with $\mathbf{x} \neq \mathbf{x}^*$. The stronger concept of **isolated local solution** requires that $\mathbf{x}^*$ is the only local solution in $\mathcal{N} \cap \Omega$.

Throughout this chapter the following concepts will play a central role.

Definition 5.1 *A point* $\mathbf{x} \in \mathbb{R}^n$ *is called* **feasible** *if* $\mathbf{x} \in \Omega$. *The* **active set** $\mathcal{A}(x)$ *at a feasible* $\mathbf{x}$ *is defined as*

$$\mathcal{A}(\mathbf{x}) = \big\{ i \in \{1, \ldots, m\} \mid g_i(\mathbf{x}) = 0 \big\}.$$

5.1.1 Two simple examples

Before deriving general optimality conditions, it is extremely helpful to look at some simple examples first.

Example 5.2 Consider a problem in two variables with a single equality constraint:

$$\min_{\mathbf{x} \in \mathbb{R}^2} x_1 + x_2 \quad \text{subject to} \quad x_1^2 + x_2^2 - 2 = 0. \tag{5.4}$$

The feasible set is a circle of radius $\sqrt{2}$ centered at 0. The solution of (5.4) is given by $(-1, -1)^T$. Geometrically, this can be seen by choosing a line $x_1 + x_2 \equiv \text{const}$ such that it touches the circle in the bottom left corner. Another (slightly less obvious) way to see this is to choose any other point on the circle and observe that we can always move along the circle (i.e., we stay feasible) and decrease the target function value at the same time. From Figure 5.2, we see that the constraint normal ∇h (with $h(\mathbf{x}) = x_1^2 + x_2^2 - 2$ is parallel to ∇f (with $f(\mathbf{x}) = x_1 + x_2$) at the solution $\mathbf{x}^*$. In other words, there is a scalar λ^* such that

$$\nabla f(\mathbf{x}^*) = \lambda^* \nabla h(\mathbf{x}^*). \tag{5.5}$$

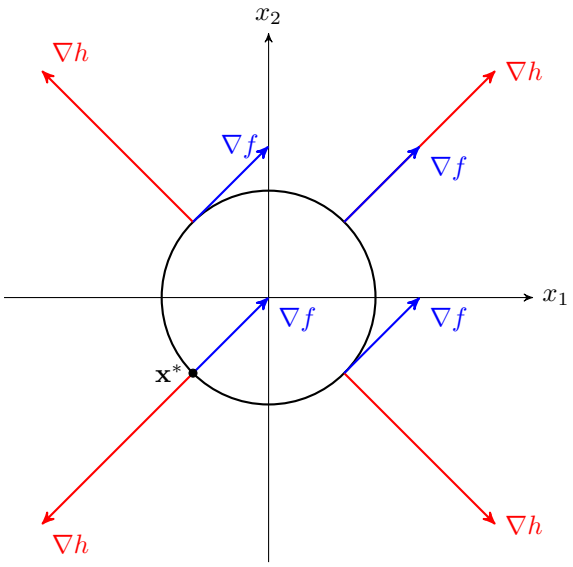


Figure 5.2. *Constraint and target function gradients for* (5.4).

By introducing the *Lagrangian function*

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda h(\mathbf{x}),$$

we can state (5.5) equivalently as follows: At the solution $\mathbf{x}^*$ there is a scalar λ^* such that

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) = 0. \tag{5.6}$$

This suggests that we can solve equality-constrained optimization problem by adapting the methods from Chapter 4 to find stationary points of $\mathcal{L}$. Of course, having a point $\mathbf{x}$ that satisfies (5.6) does in general *not* imply that $\mathbf{x}$ is a local solution. For example, (5.6) is also satisfied at $\mathbf{x} = (1, 1)^T$ with $\lambda = 1/2$. ◇

Example 5.3 In this example, we turn the equality constraint in (5.4) into an inequality constraint:

$$\min_{\mathbf{x} \in \mathbb{R}^2} x_1 + x_2 \quad \text{subject to} \quad 2 - x_1^2 - x_2^2 \geq 0. \tag{5.7}$$

Again, the solution is given by $(-1, -1)^T$. The feasible set is now the circle we had before *and* its interiour. Note that the constraint normal ∇g (with $g(\mathbf{x}) = 2 - x_1^2 - x_2^2$) now points into the interior of the feasible set at every point on the boundary, see Figure 5.3. Suppose now that $\mathbf{x}$ is feasible and we are looking for a step $\mathbf{x} \mapsto \mathbf{x} + \mathbf{s}$ such that f is decreased and the inequality constraint is still met. In first order, these two conditions on $\mathbf{s}$ amount to

$$\nabla f(\mathbf{x})^T \mathbf{s} < 0, \qquad 0 \leq g(\mathbf{x} + \mathbf{s}) \approx g(\mathbf{x}) + \nabla g(\mathbf{x})^T \mathbf{s}. \tag{5.8}$$

In order to find such a step we have to discriminate between two cases.

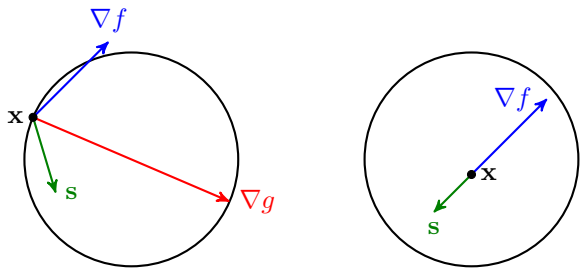


Figure 5.3. *Two feasible points* $\mathbf{x}$ *for* (5.7) *with possible improvement directions* $\mathbf{s}$.

Case of inactive g**.** If $\mathbf{x}$ is in the interior of the circle, we have $g(\mathbf{x}) > 0$. Consequently, any sufficiently small step $\mathbf{s}$ will guarantee the feasibility of $\mathbf{x}+\mathbf{s}$. Provided that $\nabla f(\mathbf{x}) \neq 0$, *any* step

$$\mathbf{s} = -\alpha \nabla f(\mathbf{x})$$

will satisfy (5.8) with $\alpha > 0$ sufficiently small.

Case of active g**.** If $\mathbf{x}$ is on the boundary of the circle, we have $g(\mathbf{x}) = 0$. The conditions (5.8) then become

$$\nabla f(\mathbf{x})^T \mathbf{s} < 0, \qquad \nabla g(\mathbf{x})^T \mathbf{s} \geq 0. \tag{5.9}$$

Both conditions define open/closed half-spaces of $\mathbb{R}^2$, see Figure 5.4. Their intersection is non-empty, unless $\nabla f(\mathbf{x})$ and $\nabla g(\mathbf{x})$ point in the same direction. The latter happens when

$$\nabla f(\mathbf{x}) = \alpha \nabla g(\mathbf{x}), \qquad \alpha \geq 0. \tag{5.10}$$

The optimality conditions for both cases, g inactive and g active, can again be summarized by using the Lagrangian

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x}).$$

At some point $\mathbf{x}^*$ there is *no* feasible step in the sense of conditions (5.8) if

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*) = 0 \tag{5.11}$$

for some $\lambda^* \geq 0$ satisfying

$$\lambda^* \cdot g(\mathbf{x}^*) = 0. \tag{5.12}$$

A condition of the form (5.12) is called *complementarity* condition. If $g(\mathbf{x}^*) \neq 0$ (inactive g), it implies that $\lambda^* = 0$ and hence (5.11) comes down to the definition $\nabla f(\mathbf{x}^*) = 0$ of a stationary point. If $g(\mathbf{x}^*) = 0$ (active g), (5.12) is satisfied for any $\lambda^* \geq 0$. In this case, (5.11) is identical with (5.10). ◇

In Example 5.3, we have seen for a single inequality constraint that it is important to discriminate between the two situations when the constraint is active and when it is inactive. This illustrates the importance of active sets from Definition 5.1.

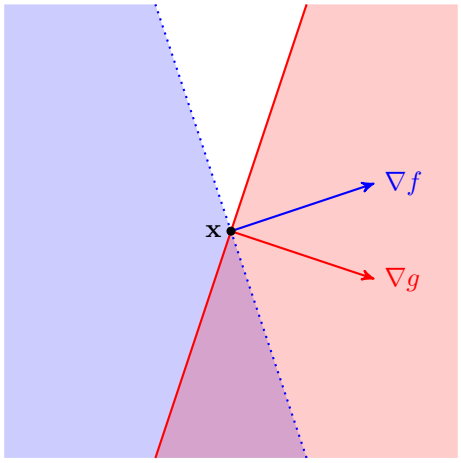


Figure 5.4. *Geometric illustration of* (5.9).

5.1.2 First-order necessary conditions

To develop optimality conditions for constrained optimization, we need to take the geometry of Ω induced by the constraints into account.

Definition 5.4 *The* **tangent cone** *of a set* $\Omega \subset \mathbb{R}^n$ *at* $\mathbf{x} \in \Omega$ *is defined as*

$$T_\Omega(\mathbf{x}) = \left\{ \mathbf{d} \in \mathbb{R}^n \mid \exists \eta_k > 0, \mathbf{x}_k \in \Omega : \ \eta_k \overset{k\to\infty}{\to} 0, \ \mathbf{x}_k \overset{k\to\infty}{\to} \mathbf{x}, \ \frac{1}{\eta_k}(\mathbf{x}_k - \mathbf{x}) \overset{k\to\infty}{\to} \mathbf{d} \right\}.$$

Put in words, Definition 5.4 means that $\mathbf{d}$ is in the tangent cone if it can be written as the limit of finite difference quotients $\frac{\mathbf{x}_k - \mathbf{x}}{\eta_k}$ along the feasible set. It is an easy exercise to see that $T_\Omega(\mathbf{x})$ is indeed a cone.[4]

Definition 5.4 directly leads to the following necessary condition.

Theorem 5.5 *If* $\mathbf{x}^*$ *is a local solution of* (5.1) *then*

$$\nabla f(\mathbf{x}^*)^T \mathbf{d} \geq 0 \qquad \forall \mathbf{d} \in T_\Omega(\mathbf{x}^*). \tag{5.13}$$

Proof. Consider $\mathbf{d} \in T_\Omega(\mathbf{x}^*)$. Then there are sequences $\mathbf{x}_k \to \mathbf{x}^*$ and $\eta_k > 0$ such that $\mathbf{d}_k := \frac{1}{\eta_k}(\mathbf{x}_k - \mathbf{x}^*) \to \mathbf{d}$. Since $\mathbf{x}^*$ is a local solution, we have $f(\mathbf{x}_k) - f(\mathbf{x}^*) \geq 0$ for sufficiently large k. Hence, it follows from the Taylor expansion that

$$\begin{aligned} 0 \leq \frac{1}{\eta_k}(f(\mathbf{x}_k) - f(\mathbf{x}^*)) &= \frac{1}{\eta_k} \nabla f(\mathbf{x}^*)^T (\mathbf{x}_k - \mathbf{x}^*) + \frac{1}{\eta_k} o(\|\mathbf{x}_k - \mathbf{x}^*\|) \\ &= \nabla f(\mathbf{x}^*)^T \mathbf{d}_k + o(1) \overset{k\to\infty}{\to} \nabla f(\mathbf{x}^*)^T \mathbf{d}, \end{aligned}$$

which completes the proof. ∎

[4] Recall that a set K is called a cone if $k \in K$ implies $\lambda k \in K$ for all $\lambda > 0$.

Theorem 5.5 is mathematically very elegant: The condition (5.13) is brief and does not depend on the particular parametrization of $T_\Omega(\mathbf{x}^*)$. Unfortunately, this implicit nature also makes it very hard to work with (5.13). It would be more practical to have conditions that are explicit in g and h. For this purpose, we need to linearize Definition 5.4. For the rest of the section we assume g and h to be continuously differentiable.

Definition 5.6 *The set of* **linearized feasible directions** *of a feasible point* $\mathbf{x} \in \Omega$ *is defined as*

$$\mathcal{F}(\mathbf{x}) = \left\{\mathbf{d} \in \mathbb{R}^n \mid \nabla g_i(\mathbf{x})^T \mathbf{d} \geq 0 \ \forall i \in \mathcal{A}(\mathbf{x}), h'(\mathbf{x})\mathbf{d} = 0\right\}.$$

The hope is that the two cones $T_\Omega(\mathbf{x})$ and $\mathcal{F}(\mathbf{x})$ are identical. In many cases, this is reasonable to expect. However, one can also construct counterexamples quite easily.

Example 5.7 We consider the non-optimal point $\mathbf{x}_0 = (-\sqrt{2}, 0)^T$ for the equality-constrained problem (5.4) from Example 5.2, that is, $f(\mathbf{x}) = x_1 + x_2$ and $h(\mathbf{x}) = x_1^2 + x_2^2 - 2$.

By considering the two directions on the circle $\mathbf{x}_0$ can be approached, one computes $T_\Omega(\mathbf{x}_0) = \{(0, d_2)^T \mid d_2 \in \mathbb{R}\}$. On the other hand, we have $\mathbf{d} \in \mathcal{F}(\mathbf{x}_0)$ if

$$0 = h'(\mathbf{x})\mathbf{d} = -2\sqrt{2}d_1.$$

Hence, $\mathcal{F}(\mathbf{x}_0)$ is identical with $T_\Omega(\mathbf{x}_0)$.

The situation changes if we replace $h(\mathbf{x})$ by $h(\mathbf{x}) = (x_1^2 + x_2^2 - 2)^2$. This still describe the same feasibility set but now

$$0 = h'(\mathbf{x})\mathbf{d} = 0,$$

that is, no constraint is imposed on $\mathbf{d}$. Hence, $\mathcal{F}(\mathbf{x}_0) = \mathbb{R}^2$ now differs from $T_\Omega(\mathbf{x}_0)$! ◇

There are many flavors of so called *constraint qualifications* that impose a condition to guarantee $T_\Omega(\mathbf{x}) = \mathcal{F}(\mathbf{x})$ at a feasible point $\mathbf{x}$. We will work with the following, relatively strong condition.

Definition 5.8 (LICQ) *We say that the* **linear independence constraint qualification (LICQ)** *holds at* $\mathbf{x} \in \Omega$ *if the active constraint gradients*

$$\{\nabla g_i(\mathbf{x}) \mid i \in \mathcal{A}(\mathbf{x})\} \cup \{\nabla h_i(\mathbf{x}) \mid i \in \{1, \ldots, p\}\}$$

form a linearly independent set.

Lemma 5.9 *Let $\mathbf{x}^\star \in \Omega$. Then:*

1. *$T_\Omega(\mathbf{x}^\star) \subset \mathcal{F}(\mathbf{x}^\star)$.*
2. *If LICQ holds at $x^\star$ then $T_\Omega(\mathbf{x}^\star) = \mathcal{F}(\mathbf{x}^\star)$.*

Proof. 1. Let $\mathbf{d} \in T_\Omega(\mathbf{x}^*)$ and consider the corresponding sequences $\mathbf{x}_k \to \mathbf{x}^*$ and $\eta_k > 0$ such that $\frac{1}{\eta_k}(\mathbf{x}_k - \mathbf{x}^*) \to \mathbf{d}$. Clearly, we have

$$\mathbf{x}_k = \mathbf{x}_* + \eta \mathbf{d} + o(\eta_k).$$

Taking into account that the equality constraints $h_1, \ldots, h_p$ are satisfied for $\mathbf{x}_k$, we obtain from the Taylor expansion that

$$0 = \frac{1}{\eta_k} h_i(\mathbf{x}_k) = \frac{1}{\eta_k} \left(h_i(\mathbf{x}^*) + \eta_k \nabla h_i(\mathbf{x}^*)^T \mathbf{d} + o(\eta_k) \right) = \nabla h_i(\mathbf{x}^*)^T \mathbf{d} + o(1).$$

Hence, $\nabla h_i(\mathbf{x}^*)^T \mathbf{d} = 0$ follows from taking the limit $k \to \infty$. In the same way, we obtain $\nabla g_i(\mathbf{x}^*)^T \mathbf{d} \geq 0$ for $i \in \mathcal{A}(\mathbf{x}^*)$.

2. By simply dropping inactive constraints beforehand, we may assume without loss of generality that all inequality constraints are active, that is, $\mathcal{A}(\mathbf{x}) = \{1, \ldots, m\}$. Then the matrix which collects all gradients,

$$A = \begin{pmatrix} g'(\mathbf{x}^\star) \\ h'(\mathbf{x}^\star) \end{pmatrix} \in \mathbb{R}^{(m+p)\times n},$$

has full row rank.[5] Now, let $Z \in \mathbb{R}^{n\times(n-m-p)}$ be a basis for the null space of A, that is, Z has full column rank and $AZ = 0$.

We now consider the (parametrized) nonlinear system of equations

$$R(\mathbf{x}, \eta) := \begin{pmatrix} g(\mathbf{x}) - \eta g'(\mathbf{x}^\star)\mathbf{d} \\ h(\mathbf{x}) - \eta h'(\mathbf{x}^\star)\mathbf{d} \\ Z^T(\mathbf{x} - \mathbf{x}^* - \eta \mathbf{d}) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{5.14}$$

with an arbitrary choice of $\mathbf{d} \in \mathcal{F}(\mathbf{x}^*)$. For $\eta = 0$, the Jacobian of R at $\mathbf{x}^\star$ is given by

$$\frac{\partial}{\partial \mathbf{x}} R(\mathbf{x}^*, 0) = \begin{pmatrix} g'(\mathbf{x}^\star) \\ h'(\mathbf{x}^\star) \\ Z^T \end{pmatrix}.$$

The construction of Z implies that this Jacobian is nonsingular. Hence, by the implicit function theorem, for all η sufficiently small the nonlinear system (5.14) has a solution $\mathbf{x}$ (which is even the unique solution in some neighborhood of $\mathbf{x}^*$) that depends continuously differentiable on η.

We now consider a sequence $\eta_k \to 0$ with $\eta_k > 0$ along with the corresponding solution $\mathbf{x}_k$ of (5.14). Then the first two equations of (5.14) imply that $\mathbf{x}_k$ is feasible. Now, by the Taylor expansion

$$0 = R(\mathbf{x}_k, \eta_k) = \left[\frac{\partial}{\partial \mathbf{x}} R(\mathbf{x}^*, 0) \right] (\mathbf{x}_k - \mathbf{x}^* - \eta_k \mathbf{d}) + o(\|\mathbf{x}_k - \mathbf{x}^*\|).$$

[5] A small subtlety: this seems to imply $m + p \leq n$, which need not to be the case in (5.1). Since we have thrown away all inactive constraints, the m in the proof might be smaller than the original m, so keeping the notation is convenient but slightly abusive.

Using the nonsingularity of the Jacobian and dividing by η_k gives

$$\frac{1}{\eta_k}(\mathbf{x}_k - \mathbf{x}^*) = \mathbf{d} + o\left(\frac{\|\mathbf{x}_k - \mathbf{x}^*\|}{\eta_k}\right),$$

from which it follows that $\mathbf{d} \in T_\Omega(\mathbf{x}^*)$. (Note that the limit on the left side exists since $\eta \mapsto \mathbf{x}(\eta)$ is differentiable in zero.) □

We have now collected all ingredients to carry over the machinery from linear programming to the nonlinear case. We recall the following central result from the course on discrete optimization.

Lemma 5.10 (Farkas' lemma) *Let $A_g \in \mathbb{R}^{n\times m}, A_h \in \mathbb{R}^{n\times p}, \mathbf{c} \in \mathbb{R}^n$. Then the following two statements are equivalent.*

1. *For any $\mathbf{d} \in \mathbb{R}^n$ with $A_g^T\mathbf{d} \geq 0$ and $A_h^T\mathbf{d} = 0$ we have $\mathbf{c}^T\mathbf{d} \geq 0$.*
2. *There exists $\mathbf{u} \in \mathbb{R}^m$ with $\mathbf{u} \geq 0$ and $\mathbf{v} \in \mathbb{R}^p$ such that $\mathbf{c} = A_g\mathbf{u} + A_h\mathbf{v}$.*

Proof. See Theorem 2.9 in [E]. □

Let us now consider a local solution $\mathbf{x}^*$ of (5.1) for which the LICQ holds. By Lemma 5.9, $T_\Omega(\mathbf{x}^\star) = \mathcal{F}(\mathbf{x}^\star)$. Thus, Theorem 5.5 implies

$$\nabla f(\mathbf{x}^*)^T\mathbf{d} \geq 0 \qquad \forall \mathbf{d} \in \mathcal{F}(\mathbf{x}^\star).$$

This is the first statement of Farkas' lemma where the columns of A_g contain all gradients $\nabla g_i(\mathbf{x}^*)$ with active inequality constraints, $A_h = h'(\mathbf{x}^*)^T$, and $\mathbf{c} = \nabla f(\mathbf{x}^*)$. The second, equivalent statement of Farkas' lemma then yields the existence of $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^p$ such that

$$\nabla f(\mathbf{x}^*) - \sum_{i\in\mathcal{A}(\mathbf{x}^*)} \lambda_i^* \nabla g_i(\mathbf{x}^*) + \sum_{i=1}^{p} \mu_i^* \nabla h_i(\mathbf{x}^*) = 0. \tag{5.15}$$

Note that the values of $\lambda_i^\star$ for $i \notin \mathcal{A}(\mathbf{x}^*)$ do not enter (5.15). Choosing these "missing" values to be zero is equivalent to the complementarity condition

$$\lambda_i g_i(\mathbf{x}^*) = 0, \qquad i = 1, \ldots, m. \tag{5.16}$$

This also allows us to rewrite (5.15) more compactly as

$$\nabla f(\mathbf{x}^*) - g'(\mathbf{x}^*)^T\lambda^* - h'(\mathbf{x}^*)^T\mu^* = 0.$$

The left-hand side turns out to be the gradient with respect to $\mathbf{x}$ at $(\mathbf{x}^*, \lambda^*, \mu^*)$ of the **Lagrangian function**

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) - \sum_{i=1}^{p} \mu_i h_i(\mathbf{x}). \tag{5.17}$$

The following theorem is the main result of this section and summarizes our findings.

Theorem 5.11 (First-order necessary conditions) *Suppose that $\mathbf{x}^*$ is a local solution of the constrained optimization problem (5.1) with continuously differentiable f, g, h, such that the LICQ holds. Then there are Lagrange multipliers λ^*, μ^* such that the following conditions are satisfied:*

(1) $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \lambda^*, \mu^*) = 0$;

(2) $h(\mathbf{x}^*) = 0$;

(3a) $g(\mathbf{x}^*) \geq 0$;

(3b) $\lambda^* \geq 0$;

(3c) $\lambda_i^* g_i(\mathbf{x}^*) = 0$ *for* $i = 1, \ldots, m$.

The conditions of Theorem 5.11 are known as **Karush-Kuhn-Tucker conditions** or, short, **KKT conditions**.

5.1.3 Special cases of the KKT conditions*

The constrained optimization problem (5.1) is called **convex** if the function f is convex, the functions g_i, $i = 1, \ldots, m$, are concave and h is affine linear. This implies that the feasible set is convex because

$$g_i(\beta\mathbf{x} + (1-\beta)\mathbf{y}) \geq \beta g_i(\mathbf{x}) + (1-\beta) g_i(\mathbf{y}) \geq 0,$$
$$h(\beta\mathbf{x} + (1-\beta)\mathbf{y}) = \beta h(\mathbf{x}) + (1-\beta) h(\mathbf{y}) = 0$$

holds for all $\mathbf{x}, \mathbf{y} \in \Omega$ and $\beta \in [0,1]$. More importantly, the KKT conditions turn out to be not only necessary but also *sufficient* for convex problems.

Theorem 5.12 *Let (5.1) be convex. Then the following statements hold:*

1. *Every local solution $\mathbf{x}^* \in \Omega$ is a global solution of (5.1).*
2. *If $\mathbf{x}^*$ satisfies the KKT conditions of Theorem 5.11 then $\mathbf{x}^*$ is a global solution of (5.1).*

Proof. 1. Let $\mathbf{x}^* \in \Omega$ be a local solution of (5.1) and consider an arbitrary $\mathbf{x} \in \Omega$. Then $\mathbf{x}^* + \beta\mathbf{d} \in \Omega$ for $\mathbf{d} = \mathbf{x} - \mathbf{x}^*$ and $\beta \in [0,1]$. By the convexity of f, it follows for sufficiently small $\beta > 0$ that

$$0 \leq f(\mathbf{x}^* + \beta\mathbf{d}) - f(\mathbf{x}^*) \leq (1-\beta) f(\mathbf{x}^*) + \beta f(\mathbf{x}) - f(\mathbf{x}^*) = \beta\big(f(\mathbf{x}) - f(\mathbf{x}^*)\big).$$

This implies $f(\mathbf{x}) - f(\mathbf{x}^*) \geq 0$ and, hence, $\mathbf{x}^*$ is a global solution.

2. Let $\mathbf{x}^* \in \Omega$ satisfy the KKT conditions, consider an arbitrary $\mathbf{x} \in \Omega$, and let $\mathbf{d} = \mathbf{x} - \mathbf{x}^*$. Then

$$\lambda_i^\star \nabla g_i(\mathbf{x})^T \mathbf{d} \geq \lambda_i^\star (g_i(\mathbf{x}) - g_i(\mathbf{x}^*)) = \lambda_i^* g_i(\mathbf{x}) \geq 0,$$

where the first inequality follows from the concavity of g_i. Moreover, $\nabla h(\mathbf{x}^*)^T \mathbf{d} = h(\mathbf{x}) - h(\mathbf{x}^*) = 0$. The convexity of f, together with the KKT conditions (1)+(3a)+(3b) yields

$$f(\mathbf{x}) - f(\mathbf{x}^*) \geq \nabla f(\mathbf{x}^*)^T \mathbf{d} = (\lambda^*)^T g'(\mathbf{x}^*)\mathbf{d} + (\mu^*)^T h'(\mathbf{x}^*)\mathbf{d} = (\lambda^*)^T g'(\mathbf{x}^*)\mathbf{d} \geq 0,$$

which completes the proof. □

In particular, Theorem 5.12 applies to the linear program

$$\min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{c}^T \mathbf{x} \quad \text{subject to} \quad \mathbf{x} \geq 0,\ A\mathbf{x} = \mathbf{b}, \tag{5.18}$$

which we will call the **primal problem**.[6] The Lagrangian (5.17) is then given by

$$\mathcal{L}(\mathbf{x}, \lambda, \mu) = \mathbf{c}^T \mathbf{x} - \mathbf{x}^T \lambda - (A\mathbf{x} - \mathbf{b})^T \mu.$$

In principle, the constraint qualification LICQ requires the matrix $\binom{E}{A}$ to have full row rank, where the rows of E consist of all unit vectors e_i^T with $i \in \mathcal{A}(x)$. However, it turns out that LICQ is in fact not needed. $\mathcal{F}$ arises from T_Ω by linearization, but the constraints in (5.18) are already linear and hence both cones must be identical.

For (5.18), the KKT conditions (which are equivalent and sufficient for a local solution) become

(p1) $A^T \mu^\star + \lambda^\star = \mathbf{c}$;

(p2) $A\mathbf{x}^\star = \mathbf{b}$;

(p3a) $\mathbf{x}^\star \geq 0$;

(p3b) $\lambda^\star \geq 0$;

(p3c) $\lambda_i^\star x_i^\star = 0$ for $i = 1, \ldots, m$.

Given the nonnegativity of $\mathbf{x}^\star, \lambda^\star$, condition (p3c) is equivalent to

$$(\mathbf{x}^\star)^T \lambda^\star = 0.$$

With the same data as in (5.18), we can define the **dual problem**

$$\max_{\mathbf{x} \in \mathbb{R}^n} \mathbf{b}^T \lambda \quad \text{subject to} \quad A^T \lambda \leq \mathbf{c}. \tag{5.19}$$

If we apply Theorem 5.11 to this problem, then the resulting KKT conditions turn out to be identical! In particular, the optimal Lagrange multipliers in the primal problem are the optimal variables in the dual problem, while the optimal Lagrange multipliers in the dual problem are the optimal variables in the primal problem. We refer to Chapter 4 of [4] for more relations between the primal and dual problems.

[6] This linear program actually takes the form of the dual problem considered in (4.2) of [E]. This nuisance is hopefully considered minor by the reader.

5.2 Quadratic Programming

A quadratic program (QP) takes the form

$$\begin{array}{rl} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{x}^T\mathbf{h} \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b} \\ & C^T\mathbf{x} \geq \mathbf{d}, \end{array} \tag{5.20}$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric and $A \in \mathbb{R}^{n\times p}$, $B \in \mathbb{R}^{n\times m}$.

We focus on this problem partly to make our life simpler, and partly because it plays an important role in the SQP method to be discussed in Section 5.3.

If G is positive semidefinite then (5.20) is convex and Theorem 5.12 applies.

5.2.1 Equality-Constrained QPs

In the absence of inequality constraints, (5.20) becomes

$$\begin{array}{rl} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{x}^T\mathbf{h} \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b}. \end{array} \tag{5.21}$$

Only the KKT conditions (1)+(2) are relevant, and they can be compactly expressed in the form of the linear system

$$\begin{pmatrix} G & -A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}^* \\ \mu^* \end{pmatrix} = \begin{pmatrix} -\mathbf{h} \\ \mathbf{b} \end{pmatrix}. \tag{5.22}$$

One disadvantage of this formulation is that the matrix in (5.22) is not symmetric. This can be fixed in several ways, for example

$$\begin{pmatrix} G & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} -\mathbf{x}^* \\ \mu^* \end{pmatrix} = \begin{pmatrix} \mathbf{h} \\ -\mathbf{b} \end{pmatrix}. \tag{5.23}$$

The linear system (5.23) (and consequently also (5.22)) has a unique solution if and only if the involved matrix is invertible.

Lemma 5.13 *Let A have full column rank. Let the columns of Z be a basis for the null space of A^T and assume that Z^TGZ is positive definite. Then the so called* **KKT matrix**

$$K = \begin{pmatrix} G & A \\ A^T & 0 \end{pmatrix}$$

is invertible.

Proof. Suppose that $K\left(\begin{smallmatrix}\mathbf{x}\\ \mathbf{y}\end{smallmatrix}\right) = 0$. Then $G\mathbf{x} = -A\mathbf{y}$ and $A^T\mathbf{x} = 0$. The latter relation states that $\mathbf{x}$ is in the null space of A^T and hence there is a vector $\mathbf{z}$ such that $\mathbf{x} = Z\mathbf{z}$. Inserting this into the first relation gives $\mathbf{z}^T Z^T \mathbf{G} Z\mathbf{z} = -\mathbf{z}^T Z^T A\mathbf{y} = 0$. From the positive definitness of $Z^T\mathbf{G}Z$ it follows that $\mathbf{z} = 0$ and therefore also $\mathbf{x} = 0$. Finally, the full column rank of A combined with $A\mathbf{y} = -G\mathbf{x} = 0$ yields $\mathbf{y}$. This shows that the null space of K is $\{0\}$. In other words, K is invertible. ∎

The positive definiteness of the reduced Hessian $Z^T G Z$ makes (5.21) essentially convex. Note that it is not required that the Hessian G itself is positive definite. A variant of Theorem 5.12 applies: One can show that the vector $\mathbf{x}^*$ satisfying (5.22) is the unique global solution of (5.21), provided that the conditions of Lemma 5.13 hold. See Theorem 16.2 in [NW].

It is important to note that the matrix K is always indefinite, unless A vanishes. We can therefore not apply solvers for symmetric positive definite linear systems (e.g., Cholesky factorization, CG method) directly for K, even when G is symmetric positive definite. However, similar to the technique used in the proof of Lemma 5.13 we can reduce this linear system to the matrix $Z^T \mathbf{G} Z$, to which, e.g., the CG method can be applied.

5.2.2 Active set methods

We now consider the general QP (5.20), for which the KKT conditions take the form

(1) $G\mathbf{x}^* - A\mu^* - C\lambda^* = -\mathbf{h}$;

(2) $A^T\mathbf{x}^* = \mathbf{b}$;

(3a) $C^T\mathbf{x}^* \geq \mathbf{d}$;

(3b) $\lambda^* \geq 0$;

(3c) $\lambda^* \odot (C^T\mathbf{x}^* - \mathbf{d}) = 0$;

where $\odot$ denotes the elementwise product. As already explained in Section 5.1.3, there is no need to require the LICQ, due to the linearity of the constraints.

In the following, we will only consider the case that G **is positive semidefinite**. This implies that (5.20) is convex and Theorem 5.12 applies. *If* the active set $\mathcal{A}(\mathbf{x}^*)$ was known, one could solve the KKT conditions above with the method discussed in Section 5.2.1 by including the active inequality constraints into the equality constraints and ignoring the inactive inequality constraints. *Active set methods* leverage this observation by maintaining a working set $\mathcal{W}_k \subset \{1, \ldots, m\}$ that satisfies $\mathcal{W}_k \subseteq \mathcal{A}(\mathbf{x}_k)$ for the kth iterate $\mathbf{x}_k$.

Given a feasible iterate $\mathbf{x}_k$ and $\mathcal{W}_k$, active set methods proceed by considering the subproblem

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{x}^T\mathbf{h} \\ \text{subject to} \quad & A^T\mathbf{x} = \mathbf{b} \\ & C_{\mathcal{W}_k}^T\mathbf{x} = \mathbf{d}_{\mathcal{W}_k}, \end{aligned} \tag{5.24}$$

where $C_{\mathcal{W}_k}$ contains the rows of C corresponding to $\mathcal{W}_k$ and $\mathbf{d}_{\mathcal{W}_k} \in \mathbb{R}^{\#\mathcal{W}_k}$ contains the corresponding entries of $\mathbf{d}$. We reformulate the minimization problem (5.24) by defining the step

$$\mathbf{p}_k = \mathbf{x} - \mathbf{x}_k,$$

which is given as the solution of

$$\begin{array}{rl} \min\limits_{\mathbf{p}\in\mathbb{R}^n} & f(\mathbf{x}) = \frac{1}{2}\mathbf{p}^T G\mathbf{p} + \mathbf{p}^T(\mathbf{h} + G\mathbf{x}_k) \\ \text{subject to} & A^T\mathbf{p} = \mathbf{b} - A^T\mathbf{x}_k \\ & C_{\mathcal{W}_k}^T\mathbf{p} = \mathbf{d}_{\mathcal{W}_k} - C_{\mathcal{W}_k}^T\mathbf{x}_k, \end{array} \tag{5.25}$$

where we dropped constant terms in the object function. According to Section 5.2.1, see (5.23), the subproblem (5.25) can be addressed by solving the linear system

$$\begin{pmatrix} G & A & C_{\mathcal{W}_k} \\ A^T & 0 & 0 \\ C_{\mathcal{W}_k}^T & 0 & 0 \end{pmatrix} \begin{pmatrix} -\mathbf{p}_k \\ \mu_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{h} + G\mathbf{x}_k \\ A^T\mathbf{x}_k - \mathbf{b} \\ C_{\mathcal{W}_k}^T\mathbf{x}_k - \mathbf{d}_{\mathcal{W}_k} \end{pmatrix}. \tag{5.26}$$

By Lemma 5.13, we need to make sure that $\begin{pmatrix} A & C_{\mathcal{W}_k} \end{pmatrix}$ has full column rank. Strategies that guarantee this property are discussed in Section 16.5 of [NW].

Given the solution $\mathbf{p}_k$ of (5.26), the updated vector $\mathbf{x}_k + \mathbf{p}_k$ solves (5.24). More importantly,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$$

satisfies the constraints of (5.24) for any $\alpha_k \in \mathbb{R}$. Because $\mathbf{x}_k$ is feasible, it is always possible to choose α_k to be the largest value in $[0, 1]$ such that $\mathbf{x}_{k+1}$ is also feasible. Considering the ith row of the relation $C^T\mathbf{x} \geq \mathbf{d}$, it is clear that we only need to worry about rows for which $i \notin \mathcal{W}_k$ and $\mathbf{c}_i^T\mathbf{p}_k < 0$. It follows that

$$\alpha_k := \min\left\{1, \min_{\substack{i\notin\mathcal{W}_k \\ \mathbf{c}_i^T\mathbf{p}_k<0}} \frac{\mathbf{d}_i - \mathbf{c}_i^T\mathbf{x}_k}{\mathbf{c}_i^T\mathbf{p}_k}\right\}. \tag{5.27}$$

If $\alpha_k < 1$ then the step $\mathbf{p}_k$ is blocked by at least one constraint j not contained in $\mathcal{W}_k$. We continue the method by setting

$$\mathcal{W}_{k+1} := \mathcal{W}_k \cup \{j\}.$$

The above procedure stagnates when it encounters $\mathbf{p}_k = 0$, implying that $\mathbf{x}_k$ is already optimal with respect to the current working set $\mathcal{W}_k$. Let λ_{k+1}, μ_{k+1} denote the corresponding Lagrange multipliers, where we set the entries of λ_{k+1} to zero at indices not contained in $\mathcal{W}_k$. If $\lambda_{k+1} \geq 0$ then all KKT conditions for (5.20) are satisfied. Hence, we have found a global solution $\mathbf{x}_k$ and stop the method. If, on the other hand, there is an index $j \in \mathcal{W}_k$ with $\lambda_{k+1,j} < 0$ then this index is removed,

$$\mathcal{W}_{k+1} := \mathcal{W}_k \setminus \{j\},$$

and the method is continued. In summary, we obtain the algorithm below. Note that feasible starting points can be obtained by using techniques from linear programming, see [E,NW].

Algorithm 5.1. Active set method for convex QP.

Input: QP (5.20) with symmetric positive semidefinite G and feasible starting point $\mathbf{x}_0$.

Output: Global solution $\mathbf{x}^*$.

for $k = 0, 1, 2, \ldots$ **do**
  Compute $\mathbf{p}_k$ such that $\mathbf{p}_k + \mathbf{x}_k$ solves (5.24), along with the corresponding Lagrange multipliers λ_{k+1}, μ_{k+1}.
  if $\mathbf{p}_k = 0$ and $\lambda_{k+1} \geq 0$ **then**
    Stop with $\mathbf{x}^* = \mathbf{x}_k$.
  else if $\mathbf{p}_k = 0$ and $\lambda_{k+1} \not\geq 0$ **then**
    Choose $j \leftarrow \arg\min_{j \in \mathcal{W}_k} \lambda_{k+1,j}$.
    Set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k, \mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$.
  else
    Compute α_k as in (5.27) and set $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$.
    if $\alpha_k < 1$ **then**
      Obtain $\mathcal{W}_{k+1}$ by adding blocking constraint to $\mathcal{W}_k$.
    else
      Set $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$.
    end if
  end if
end for

One of the most remarkable properties of this algorithm is that it terminates after a *finite* number of steps, provided that $\alpha_k > 0$ for every nonzero search direction $\mathbf{p}_k$, see Pg. 157 in [NW]. In the rare case that $\alpha_k = 0$, the algorithm may run into a cycle. Tricks similar to the ones used in linear programming can be used to break the cycle.

5.2.3 Interior point methods

Together with the simplex method, interior point methods are one of the most frequently used solvers in linear programming. They have also turned out to be quite successful in addressing general nonconvex nonlinear optimization problems, see Chapter 19 in [NW]. In this lecture, we will restrict our coverage of this important class of methods to convex quadratic programs. Moreover, for simplicity, we will neglect equality constraints:

$$\begin{array}{rl} \min\limits_{\mathbf{x} \in \mathbb{R}^n} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G \mathbf{x} + \mathbf{x}^T \mathbf{h} \\ \text{subject to} & C^T \mathbf{x} \geq \mathbf{d}, \end{array} \tag{5.28}$$

where G is symmetric positive semidefinite. Let us recall that the KKT conditions take the form

(1) $G\mathbf{x} - C\lambda = -\mathbf{h}$;

(3a) $C^T \mathbf{x} \geq \mathbf{d}$;

(3b) $\lambda \geq 0$;

(3c) $\lambda \odot (C^T \mathbf{x} - \mathbf{d}) = 0$.

Thanks to convexity, these conditions are sufficient and necessary for a global solution of (5.28).

To avoid dealing with $C^T\mathbf{x} \geq \mathbf{d}$, we now introduce the so called **slack vector** $\mathbf{s} \geq 0$, yielding the equivalent conditions

(1) $G\mathbf{x} - C\lambda = -\mathbf{h}$;

(Sa) $C^T\mathbf{x} - \mathbf{s} = \mathbf{d}$;

(Sb) $\lambda \geq 0; \mathbf{s} \geq 0$;

(Sc) $\lambda \odot \mathbf{s} = 0$.

Interior point methods generate iterates $(\mathbf{x}, \lambda, \mathbf{s})$ that satisfy (Sb) *strictly*, that is, $\lambda > 0$ and $\mathbf{s} > 0$. Of course, this means that (Sc) cannot be satisfied exactly. To quantify how far we are off from complementarity, we consider the average of the entries in $\lambda \odot \mathbf{s}$ (which are all positive):

$$\mu := \frac{1}{m}\mathbf{s}^T\lambda. \tag{5.29}$$

The correspondingly *perturbed* KKT conditions can be written as a nonlinear system

$$F(\mathbf{x}, \mathbf{s}, \lambda, \sigma\mu) := \begin{pmatrix} G\mathbf{x} - C\lambda + \mathbf{h} \\ C^T\mathbf{x} - \mathbf{s} - \mathbf{d} \\ \mathcal{S}\Lambda\mathbf{e} - \sigma\mu\mathbf{e} \end{pmatrix} = 0, \tag{5.30}$$

where

$$\mathcal{S} = \operatorname{diag}(s_1, \ldots, s_m), \quad \Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_m), \quad \mathbf{e} = (1, \ldots, 1)^T.$$

The solutions of (5.30) define the so called **central path**. We aim to bring μ down to zero, while maintaining the positivity of $\mathbf{y}$ and $\mathbf{s}$ at the same time.

By considering μ fixed, Newton's method applied to (5.30) yields the linear system

$$\begin{pmatrix} G & 0 & -C \\ C^T & -I & 0 \\ 0 & \Lambda & \mathcal{S} \end{pmatrix} \begin{pmatrix} \triangle\mathbf{x} \\ \triangle\mathbf{s} \\ \triangle\lambda \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_d \\ -\mathbf{r}_p \\ -\mathcal{S}\Lambda\mathbf{e} + \sigma\mu\mathbf{e} \end{pmatrix}, \tag{5.31}$$

with

$$\mathbf{r}_d = G\mathbf{x} - C\lambda + \mathbf{h}, \qquad \mathbf{r}_p = C^T\mathbf{x} - \mathbf{s} - \mathbf{d}.$$

Given a current iterate $(\mathbf{x}_k, \mathbf{s}_k, \lambda_k)$ with $\mathbf{s}_k > 0, \lambda_k > 0$, interior point methods proceed as follows to produce the next iterate:

1. Compute $\mu = \frac{1}{m}\mathbf{s}^T\lambda$ according to (5.29).
2. Choose the barrier parameter reduction factor $\sigma \in [0, 1]$.
3. Determine the Newton direction $(\triangle\mathbf{x}, \triangle\mathbf{s}, \triangle\lambda)$ from the current iteration by solving (5.31).

4. Choose a step length $\alpha > 0$ such that $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha\triangle\mathbf{x} > 0$ and $\mathbf{s}_{k+1} := \mathbf{s}_k + \alpha\triangle\mathbf{s} > 0$.

5. Set $\lambda_{k+1} = \lambda_k + \alpha\triangle\lambda$.

The convergence speed of this iteration crucially depends on the choice of σ in Step 2 and the choice of α in Step 4. We refer to the discussion in Section 16.6 of [NW].

5.3 Sequential Quadratic Programming (SQP)*

Sequential Quadratic Programming is one of the most successful techniques in dealing with general nonlinear constrained optimization problems. It generates steps by solving quadratic programming subproblems, using one of the algorithms discussed in the previous section.

5.3.1 Local SQP for equality constraints

To start simple, we first discuss the computation of steps for the equality-constrained problem

$$\begin{array}{rl} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}) \\ \text{subject to} & h(\mathbf{x}) = 0. \end{array} \tag{5.32}$$

Let us recall that the Lagrangian for this problem is given by

$$\mathcal{L}(x, \lambda) = f(\mathbf{x}) - \mu^T h(\mathbf{x}),$$

leading to the KKT conditions

$$F(\mathbf{x}, \mu) := \begin{pmatrix} \nabla f(\mathbf{x}) - h'(\mathbf{x})^T\mu \\ h(\mathbf{x}) \end{pmatrix} = 0.$$

The Newton method applied to this nonlinear system takes the form

$$\begin{pmatrix} \mathbf{x}_{k+1} \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k \\ \mu_k \end{pmatrix} + \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k,\mu} \end{pmatrix} \tag{5.33}$$

with the correction vector satisfying the linear system

$$\begin{pmatrix} \mathsf{H}_k & -h'(\mathbf{x}_k)^T \\ h'(\mathbf{x}_k) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{p}_{k,\mu} \end{pmatrix} = \begin{pmatrix} -\nabla f(\mathbf{x}_k) + h'(\mathbf{x}_k)^T\mu_k \\ -h(\mathbf{x}_k) \end{pmatrix} \tag{5.34}$$

Here, $\mathsf{H}_k := \mathcal{L}_{\mathbf{xx}}(\mathbf{x}_k, \mu_k)$ denotes the Hessian of $\mathcal{L}$ with respect to $\mathbf{x}$ at $(\mathbf{x}_k, \mu_k)$. By Lemma 5.13, the so called Newton-KKT system (5.34) is uniquely solvable at $(\mathbf{x}, \mu)$ if

(A1) The constraint Jacobian $h'(\mathbf{x})$ has full row rank.

(A2) $\mathbf{d}^T\mathsf{H}_k\mathbf{d} > 0$ for all $\mathbf{d}$ such that $h'(\mathbf{x})\mathbf{d} = 0$.

Assumption (A1) is our good old LICQ, see Definition 5.8. For $(\mathbf{x}, \mu)$ sufficiently close to a local solution $(\mathbf{x}^*, \mu^*)$, Assumption (A2) follows from a second-order sufficiency condition for $(\mathbf{x}^*, \mu^*)$ that has been discussed in the exercises, see also [NW]. Since (A1) and (A2) guarantee the invertibility of the Jacobian, standard results for Newton methods imply local quadratic convergence to a local solution $(\mathbf{x}^*, \mu^*)$ for which LICQ and the second-order sufficiency condition hold.

We now present an alternative way to derive (5.34), which admits a more straightforward extension to inequality constraints. Around the current iterate $(\mathbf{x}_k, \mu_k)$, the nonlinear problem (5.32) is replaced by the quadratic program

$$\begin{aligned} \min_{\mathbf{p}\in\mathbb{R}^n} \quad & f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{p} + \tfrac{1}{2}\mathbf{p}^T \mathsf{H}_k \mathbf{p} \\ \text{subject to} \quad & h'(\mathbf{x}_k)\mathbf{p} + h(\mathbf{x}_k) = 0. \end{aligned} \tag{5.35}$$

Note that the constant term $f(\mathbf{x}_k)$ has no effect on the minimizer. By the discussion in Section 5.2.1, see in particular (5.22)[7], the solution $\mathbf{p}_k$ of the QP (5.35) satisfies the linear system

$$\begin{pmatrix} \mathsf{H}_k & -h'(\mathbf{x}_k) \\ h'(\mathbf{x}_k)^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f(\mathbf{x}_k) \\ -h(\mathbf{x}_k) \end{pmatrix}. \tag{5.36}$$

This linear system is uniquely solvable if assumptions (A1) and (A2) are satisfied.

Using $\mu_{k+1} = \mu_k + \mathbf{p}_{k,\mu}$, it turns out that (5.36) is identical with (5.34). Hence both approaches, (i) applying Newton to the KKT conditions and (ii) locally solving a QP, produce the same iterates!

5.3.2 Local SQP for equality and inequality constraints

The local SQP method (5.35) is easily extended to the general constrained optimization problem (5.1):

$$\begin{aligned} \min_{\mathbf{p}\in\mathbb{R}^n} \quad & \nabla f(\mathbf{x}_k)^T \mathbf{p} + \tfrac{1}{2}\mathbf{p}^T \mathsf{H}_k \mathbf{p} \\ \text{subject to} \quad & g'(\mathbf{x}_k)\mathbf{p} + g(\mathbf{x}_k) \geq 0 \\ & h'(\mathbf{x}_k)\mathbf{p} + h(\mathbf{x}_k) = 0. \end{aligned} \tag{5.37}$$

The solution $\mathbf{p}_k$ to this QP can be obtained with the methods discussed in Section 5.2, resulting in the next iterate $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{p}_k$, along with the correspondingly updated Lagrange multipliers λ_{k+1}, μ_{k+1}. It can be shown that the sequence resulting $(\mathbf{x}_k, \lambda_k, \mu_k)$ converges locally quadratically to a triple $(\mathbf{x}^*, \lambda^*, \mu^*)$ satisfying the KKT conditions.

5.3.3 Globalized SQP

To perform line search (or any other kind of strategy ensuring global convergence), we not only have to take the value of the objective function into account but also

[7] Be careful, the choice of notation in this section and in Section 5.2.1 do not match.

the satisfaction of the constraints. This is the purpose of **merit functions**. One popular choice is the ℓ_1 penalty function defined by

$$\phi_\beta(\mathbf{x}) = f(\mathbf{x}) + \beta \sum_{i=1}^{m} \max\{0, -g_i(\mathbf{x})\} + \beta \sum_{i=1}^{p} |h_i(\mathbf{x})|$$

for some parameter $\beta > 0$. Defining the vector $(g(\mathbf{x}))_-$ componentwise by $(g_i)_- = \max\{0, -g_i(\mathbf{x})\}$ and using the usual ℓ_1 vector norm, we can express this penalty function in more compact form as

$$\phi_\beta(\mathbf{x}) = f(\mathbf{x}) + \beta\|(g(\mathbf{x}))_-\|_1 + \beta\|h(\mathbf{x})\|_1. \tag{5.38}$$

Due to the presence of the maximum and the absolute value in the definition, the function ϕ_β is not differentiable everywhere. Instead, we will work with the (one-sided) directional derivative *[dérivée directionnelle au sens de Dini]*, which is defined as

$$D_+(\phi_\beta(\mathbf{x}))[\mathbf{p}] = \lim_{\substack{\alpha \to 0 \\ \alpha > 0}} \frac{\phi_\beta(\mathbf{x} + \alpha\mathbf{p}) - \phi_\beta(\mathbf{x})}{\alpha}.$$

for a direction $\mathbf{p} \in \mathbb{R}^n$. This derivative always exists and significantly simplifies if we only consider directions produced by local SQP.

Lemma 5.14 *Let f, g, h be continuously differentiable and let $\mathbf{p}_k$ satisfies the constraints of the QP* (5.37)*. Then*

$$D_+(\phi_\beta(\mathbf{x}_k))[\mathbf{p}_k] = \nabla f(\mathbf{x}_k)^T\mathbf{p}_k - \beta\|h(\mathbf{x}_k)\|_1 - \beta \sum_{g_i(\mathbf{x}_k)<0} \nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k. \tag{5.39}$$

Proof. We treat each term in (5.38) separately. Since f is differentiable, we have

$$D_+(f(\mathbf{x}_k))[\mathbf{p}_k] = \nabla f(\mathbf{x}_k)^T\mathbf{p}_k. \tag{5.40}$$

For the second term, we first note that the differentiability of g implies

$$\|(g(\mathbf{x}_k + \alpha\mathbf{p}_k))_-\|_1 - \|g(\mathbf{x}_k)\|_1 = \|(g(\mathbf{x}_k) + \alpha g'(\mathbf{x}_k)\mathbf{p}_k))_-\|_1 - \|(g(\mathbf{x}_k))_-\|_1 + o(\alpha)$$

For $g_i(\mathbf{x}_k) > 0$, we have $(g(\mathbf{x}_k))_- = (g_i(\mathbf{x}_k) + \alpha\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k))_- = 0$ for sufficiently small α and hence $D_+((g_i(\mathbf{x}_k))_-)[\mathbf{p}_k] = 0$. For $g_i(\mathbf{x}_k) < 0$, we readily obtain $D_+((g_i(\mathbf{x}_k))_-)[\mathbf{p}_k] = -\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k$. For $g_i(\mathbf{x}_k) = 0$, we have

$$(g_i(\mathbf{x}_k) + \alpha\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k))_- - (g_i(\mathbf{x}_k))_- = \alpha(\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k))_- = 0,$$

where we used $\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k \geq -g(\mathbf{x}_k) > 0$ in the last step. This follows from the constraint $g'(\mathbf{x}_k)\mathbf{p}_k + g(\mathbf{x}_k) \geq 0$ in (5.37). In summary, we obtain

$$D_+(\|(g(\mathbf{x}_k))_-\|_1)[\mathbf{p}_k] = -\sum_{g_i(\mathbf{x}_k)<0} \nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k. \tag{5.41}$$

For the third term, the differentiability of h implies

$$\begin{aligned}\|h(\mathbf{x}_k + \alpha\mathbf{p}_k)\|_1 - \|h(\mathbf{x}_k)\|_1 &= \|h(\mathbf{x}_k) + \alpha h'(\mathbf{x}_k)\mathbf{p}_k)\|_1 - \|h(\mathbf{x}_k)\|_1 + o(\alpha) \\ &= (1-\alpha)\|h(\mathbf{x}_k)\|_1 - \|h(\mathbf{x}_k)\|_1 + o(\alpha) \\ &= -\alpha\|h(\mathbf{x}_k)\|_1 + o(\alpha)\end{aligned}$$

for $\alpha \leq 1$, where we used the relation $h'(\mathbf{x}_k)\mathbf{p}_k = -h(\mathbf{x}_k)$ from the constraints in (5.37). Hence,

$$D_+(\|h(\mathbf{x}_k)\|_1)[\mathbf{p}_k] = \|h(\mathbf{x}_k)\|_1. \tag{5.42}$$

Combining (5.40), (5.41), and (5.42) concludes the proof. □

Provided that LICQ holds, a local solution of the QP will give rise to a triple $(\mathbf{p}_k, \lambda_{k+1}, \mu_{k+1})$ satisfying the KKT conditions

(1) $\mathsf{H}_k\mathbf{p}_k - g'(\mathbf{x}_k)^T\lambda_{k+1} - h'(\mathbf{x}_k)^T\mu_{k+1} = -\nabla f(\mathbf{x}_k)$;

(2) $h'(\mathbf{x}_k)\mathbf{p}_k + h(\mathbf{x}_k) = 0$;

(3a) $g'(\mathbf{x}_k)\mathbf{p}_k + g(\mathbf{x}_k) \geq 0$;

(3b) $\lambda_{k+1} \geq 0$;

(3c) $\lambda_{k+1} \odot (g'(\mathbf{x}_k)\mathbf{p}_k + g(\mathbf{x}_k)) = 0$.

As the following theorem shows, these KKT conditions together with the positive definiteness of the Hessian imply that p_k is a descent direction for the merit function ϕ_β, provided that β is sufficiently large.

Theorem 5.15 *Let f, g, h be continuously differentiable and let $(\mathbf{p}_k, \lambda_{k+1}, \mu_{k+1})$ satisfy the KKT conditions above. Then*

$$D_+(\phi_\beta(\mathbf{x}_k))[\mathbf{p}_k] \leq -\mathbf{p}_k^T\mathsf{H}_k\mathbf{p}_k,$$

provided that $\beta \geq \max\{\|\lambda_{k+1}\|_\infty, \|\mu_{k+1}\|_\infty\}$.

Proof. The KKT conditions (1) and (2) give

$$\begin{aligned}\nabla f(\mathbf{x}_k)^T\mathbf{p}_k &= -\mathbf{p}_k^T\mathsf{H}_k\mathbf{p}_k + \lambda_{k+1}^T g'(\mathbf{x}_k)\mathbf{p}_k + \mu_{k+1}^T h'(\mathbf{x}_k)\mathbf{p}_k \\ &= -\mathbf{p}_k^T\mathsf{H}_k\mathbf{p}_k + \lambda_{k+1}^T g'(\mathbf{x}_k)\mathbf{p}_k - \mu_{k+1}^T h(\mathbf{x}_k).\end{aligned}$$

The middle term is handled by the KKT conditions (3a)–(3c), which imply

$$\begin{aligned}\lambda_{k+1}^T g'(\mathbf{x}_k)\mathbf{p}_k &= \sum_{g_i(\mathbf{x}_k)<0} \lambda_{k+1,i}\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k + \sum_{g_i(\mathbf{x}_k)\geq 0} \lambda_{k+1,i}\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k \\ &= \sum_{g_i(\mathbf{x}_k)<0} \lambda_{k+1,i}\underbrace{\nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k}_{>0} - \sum_{g_i(\mathbf{x}_k)\geq 0} \lambda_{k+1,i} g_i(\mathbf{x}_k) \\ &\leq \beta \sum_{g_i(\mathbf{x}_k)<0} \nabla g_i(\mathbf{x}_k)^T\mathbf{p}_k.\end{aligned}$$

Note that we also have $-\mu_{k+1}^T h(\mathbf{x}_k) \leq \beta\|h(\mathbf{x}_k)\|_1$.

Combining these relations with the result of Lemma 5.14 gives

$$\begin{aligned} D_+(\phi_\beta(\mathbf{x}_k))[\mathbf{p}_k] &= \nabla f(\mathbf{x}_k)^T \mathbf{p}_k - \beta \|h(\mathbf{x}_k)\|_1 - \beta \sum_{g_i(\mathbf{x}_k)<0} \nabla g_i(\mathbf{x}_k)^T \mathbf{p}_k \\ &\leq -\mathbf{p}_k^T \mathsf{H}_k \mathbf{p}_k - \mu_{k+1}^T h(\mathbf{x}_k) - \beta \|h(\mathbf{x}_k)\|_1 \\ &\leq -\mathbf{p}_k^T \mathsf{H}_k \mathbf{p}_k, \end{aligned}$$

which concludes the proof. □

None of the derivations above made use of the fact that H_k is the Hessian of the Lagrangian. Hence, the result of Theorem 5.15 remains valid for an arbitrary symmetric matrix H_k. Therefore the ideas for quasi-Newton methods from Section 4.2.5 carry over to constrained optimization problems. Again, it is important to maintain the positive definitness H_k to guarantee that $\mathbf{p}_k$ is a descent direction under the conditions of Theorem 5.15.

Finally, we provide the complete algorithms

Algorithm 5.2. SQP with Armijo line search.

Input: Functions f, g, h, starting vectors $\mathbf{x}_0, \lambda_0, \mu_0$ and parameters $\alpha > 0$, $\beta > 0$, $c_1 > 0$.

Output: Vector $\mathbf{x}_k$ approximating stationary point.

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Determine a solution $\mathbf{p}_k$ of the QP (5.35) along with Lagrange multipliers λ, μ.
3: Set $\mathbf{p}_\lambda = \lambda - \lambda_k$, $\mathbf{p}_\mu = \mu - \mu_k$.
4: Determine the largest number $\alpha_k \in \{1, 2^{-1}, 2^{-2}, \ldots\}$ such that

$$\phi_\beta(\mathbf{x}_k + \alpha_k \mathbf{p}_k) - \phi_\beta(\mathbf{x}_k) \leq c_1 \alpha_k D_+(\phi_\beta(\mathbf{x}_k))[\mathbf{p}_k]$$

5: Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$, $\lambda_{k+1} = \lambda_k + \alpha_k \mathbf{p}_\lambda$, $\mu_{k+1} = \mu_k + \alpha_k \mathbf{p}_\mu$.
6: **end for**

Algorithm 5.3.3 is the most basic variant of a globalized SQP algorithm. As already mentioned above, in practice one normally uses quasi-Newton techniques (such as BFGS) to avoid the computation of the Hessian. A number of additional subtleties have to be taken into account when implementing Algorithm 5.3.3. For example, it may happen that one QP (5.35) appearing in the course of Algorithm 5.3.3 does not admit a solution, for example because the feasible set is empty. One possible way out of such a situation is to relax the constraints of the QP (5.35). We refer to Chapter 18 in [NW] for this and other details on the SQP algorithm.

5.4 Projected gradient methods

Both, the active set method and the interior point require the solution of a linear system in every step, which may become too costly for large-scale problems. Moreover, the active set method has the additional disadvantage for a large number of variables that at most one index can be inserted or removed in $\mathcal{W}_k$. Many applications, especially in statistical/machine learning, do not require high accuracy and a first-order method would be sufficient. The basic idea of *projected gradient methods* is to perform a gradient step and then project it to satisfy the constraints. To carry out the projection effectively requires the constraints to be sufficiently simple. It turns out that projected gradient methods are best understood and analyzed via proximal point mappings.[8]

5.4.1 Subgradients and proximal point mappings

Consider an *extended* function $p : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and let $\operatorname{dom} p = \{\mathbf{x} \in \mathbb{R}^n : p(x) < \infty\}$ denote the domain of p. We call the extended function p convex if $\operatorname{dom} p$ is a convex set and p is convex on $\operatorname{dom} p$ in the sense of Definition 4.18.

The role of p will be to describe a convex constraint set (see Example 5.19 below) and, as such, it will not be differentiable. The notion of subgradients is the most common way to extend the concept of gradients to non-differentiable convex functions.

Definition 5.16 *Let $p : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be convex. A* subgradient *of p at $\mathbf{x}_0 \in \operatorname{dom} p$ is any vector $g \in \mathbb{R}^n$ such that*

$$p(\mathbf{x}) \geq p(\mathbf{x}_0) + \langle g, \mathbf{x} - \mathbf{x}_0 \rangle, \qquad \forall \mathbf{x} \in \operatorname{dom} p. \tag{5.43}$$

The subdifferential $\partial p(\mathbf{x}_0)$ *is the set of all subgradients of f at $\mathbf{x}_0$.*

Subgradients have a number of important and interesting properties; see Section 3.1 of [N]. In the following, we limit ourselves to the properties that are needed for the subsequent developments.

Theorem 5.17 *Let $p : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be convex. $\mathbf{x}^\star \in \mathbb{R}^n$ is a global minimum of p if and only if $0 \in \partial p(\mathbf{x}^\star)$.*

Proof. This immediately follows from the definition: $0 \in \partial p(\mathbf{x}^\star)$ is equivalent to

$$p(\mathbf{x}) \geq p(\mathbf{x}^*) + \langle 0, \mathbf{x} - \mathbf{x}^* \rangle = p(\mathbf{x}^*)$$

for all $\mathbf{x} \in \operatorname{dom} p$. □

[8] Part of the material in this section is based on Recht's lecture notes available from `http://pages.cs.wisc.edu/~brecht/cs726docs/ProjectedGradientMethods.pdf`.

Lemma 5.18 *For* $\mathbf{x}, \mathbf{y} \in \operatorname{dom} p$ *let* $\mathbf{g_x} \in \partial p(\mathbf{x}), \mathbf{g_y} \in \partial p(\mathbf{y})$. *Then* $\langle \mathbf{g_x} - \mathbf{g_y}, \mathbf{x} - \mathbf{y} \rangle \geq 0$.

Proof. By definition,

$$\langle g_{\mathbf{x}}, \mathbf{y} - \mathbf{x} \rangle \leq p(\mathbf{y}) - p(\mathbf{x}), \quad \langle g_{\mathbf{y}}, \mathbf{x} - \mathbf{y} \rangle \leq p(\mathbf{x}) - p(\mathbf{y}).$$

Adding both inequalities yields $-\langle \mathbf{g_x} - \mathbf{g_y}, \mathbf{x} - \mathbf{y} \rangle \leq 0$. □

Given a convex function $p : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, we define the *proximity operator* or the *proximity point mapping* as

$$\operatorname{prox}_p(\mathbf{x}) := \arg\min_{\mathbf{y}} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 + p(\mathbf{y}). \tag{5.44}$$

Note that the strong convexity of the target function implies that the minimizer is uniquely defined. By Theorem 5.17, $\operatorname{prox}_p(\mathbf{x})$ is the unique vector satisfying

$$\mathbf{x} - \operatorname{prox}_p(\mathbf{x}) \in \partial p(\operatorname{prox}_p(\mathbf{x})).$$

Example 5.19 Let C be a convex set. Then the indicator function

$$i_C(\mathbf{x}) := \begin{cases} 0 & \mathbf{x} \in C, \\ \infty & \text{otherwise}, \end{cases}$$

is convex. By its definition (5.44), $\operatorname{prox}_{i_C}(\mathbf{x})$ is the point in C closest to $\mathbf{x}$.

For $p(\mathbf{x}) = \mu\|\mathbf{x}\|_1$, we have

$$\big(\operatorname{prox}_{i_C}(\mathbf{x})\big)_i = \begin{cases} x_i + \mu & \text{if } x_i < -\mu, \\ x_i - \mu & \text{if } x_i > \mu, \\ 0 & \text{otherwise.} \end{cases}$$

◇

Lemma 5.20 *Let* $q(\mathbf{x}) := \mathbf{x} - \operatorname{prox}_p(\mathbf{x})$. *Then the following relations hold:*

1. $q(\mathbf{x}) \in \partial p(\operatorname{prox}_p(\mathbf{x}))$;
2. $\langle \operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y}), q(\mathbf{x}) - q(\mathbf{y}) \rangle \geq 0$;
3. $\|\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y})\|_2^2 + \|q(\mathbf{x}) - q(\mathbf{y})\|_2^2 \leq \|\mathbf{x} - \mathbf{y}\|_2^2$;
4. $\|\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y})\|_2 = \|\mathbf{x} - \mathbf{y}\|_2$ *if and only if* $\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y}) = \mathbf{x} - \mathbf{y}$.

Proof. 1. follows by definition. 2. follows from 1. combined with Lemma 5.18. 3. follows from

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|_2^2 &= \|(\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y})) + (q(\mathbf{x}) - q(\mathbf{y}))\|_2^2 \\ &= \|\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y})\|_2^2 + 2\langle \operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y}), q(\mathbf{x}) - q(\mathbf{y}) \rangle \\ &\quad + \|q(\mathbf{x}) - q(\mathbf{y})\|_2^2 \\ &\geq \|\operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y})\|_2^2 + \|q(\mathbf{x}) - q(\mathbf{y})\|_2^2. \end{aligned}$$

where we used 2. in the inequality. 4. is a direct consequence of 3. □

Lemma 5.20.3 implies that the proximity operator is *nonexpansive*:

$$\| \operatorname{prox}_p(\mathbf{x}) - \operatorname{prox}_p(\mathbf{y}) \|_2^2 \leq \|\mathbf{x} - \mathbf{y}\|_2^2. \tag{5.45}$$

In other words, $\operatorname{prox}_p(\mathbf{x})$ is Lipschitz continuous constant 1.

Given $\mathbf{x}_0 \in \mathbb{R}^n$, the *proximal point algorithm* is defined by the iteration

$$\mathbf{x}_{k+1} = \operatorname{prox}_p(\mathbf{x}_k), \qquad k = 0, 1, \ldots. \tag{5.46}$$

Lemma 5.21 *The proximal point algorithm* (5.46) *converges to a minimizer of* p.

Proof. The nonexpansive property (5.45) implies that the sequence $\{\mathbf{x}_k\}$ is bounded and therefore has one or several limit points. Let $\bar{\mathbf{x}}$ be such a limit point and let $\mathbf{x}^\star$ be a minimizer of p, that is, $0 \in \partial p(\mathbf{x}^\star)$. Again by (5.45), we have

$$\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_2 = \| \operatorname{prox}_p(\mathbf{x}_k) - \operatorname{prox}_p(\mathbf{x}^\star)\|_2 \leq \|\mathbf{x}_k - \mathbf{x}^\star\|_2.$$

Hence

$$\|\mathbf{x}_k - \mathbf{x}^\star\|_2 \overset{k\to\infty}{\longrightarrow} \|\bar{\mathbf{x}} - \mathbf{x}^\star\|_2. \tag{5.47}$$

By continuity, $\operatorname{prox}_p(\bar{\mathbf{x}})$ is also a limit point of $\mathbf{x}_k$ and, consequently,

$$\| \operatorname{prox}_p(\bar{\mathbf{x}}) - \operatorname{prox}_p(\mathbf{x}^\star)\|_2 = \| \operatorname{prox}_p(\bar{\mathbf{x}}) - \mathbf{x}^\star\|_2 = \|\bar{\mathbf{x}} - \mathbf{x}^\star\|_2.$$

By Lemma 5.20.4, this implies $\operatorname{prox}_p(\bar{\mathbf{x}}) - \operatorname{prox}_p(\mathbf{x}^\star) = \bar{\mathbf{x}} - \mathbf{x}^\star$. Hence $\operatorname{prox}_p(\bar{\mathbf{x}}) = \bar{\mathbf{x}}$ and $0 \in \partial p(\bar{\mathbf{x}})$. This allows us to replace $\mathbf{x}^\star$ by $\bar{\mathbf{x}}$ in (5.47), leading to $\|\mathbf{x}_k - \bar{\mathbf{x}}\|_2 \overset{k\to\infty}{\longrightarrow} 0$. □

5.4.2 The projected gradient scheme

We now consider a convex optimization problem that admits a decomposition of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + p(\mathbf{x}), \tag{5.48}$$

where f is a smooth convex function, while p is an extended convex function. The idea is that p contains all the difficulty in terms of nonsmoothness but its form is sufficiently simple that it is possible to apply the proximity operator conveniently. Important examples for p are those given in Example 5.19. In principle, one can extend the gradient method to use (normalized) subgradients of $f(\mathbf{x}) + p(\mathbf{x})$, but this method can converge quite slowly if p is nonsmooth. In such cases, the following *projected gradient scheme* can be much faster:

$$\mathbf{x}_{k+1} = \operatorname{prox}_{\alpha_k p}\big(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\big), \qquad k = 0, 1, \ldots, \tag{5.49}$$

where $\alpha_0, \alpha_1, \ldots$ is a sequence of (suitably chosen) positive step sizes.

The rationale behind algorithm (5.49) follows from the following simple result.

Lemma 5.22 *Let f, p be convex and additionally assume that f is differentiable. Then $\mathbf{x}^\star$ is a minimizer for (5.48) if and only if*

$$\mathbf{x}^\star = \operatorname{prox}_{\alpha p}\left(\mathbf{x}^\star - \alpha \nabla f(\mathbf{x}^\star)\right)$$

holds for all $\alpha > 0$.

Proof. By Theorem 5.17, $\mathbf{x}^\star$ is a minimizer if and only if

$$-\nabla f(\mathbf{x}^\star) \in \partial p(\mathbf{x}^\star) \quad \Leftrightarrow \quad \mathbf{x}^\star - \alpha \nabla f(\mathbf{x}^\star) - \mathbf{x}^\star \in \alpha \partial p(\mathbf{x}^\star),$$

which is equivalent to $\mathbf{x}^\star = \operatorname{prox}_{\alpha p}\left(\mathbf{x}^\star - \alpha \nabla f(\mathbf{x}^\star)\right)$. ∎

In the convergence analysis of (5.49) we focus on the case of an L-smooth μ-strongly convex function f. Recall that this implies

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}\|\mathbf{y} - \mathbf{x}\|_2^2. \tag{5.50}$$

Since $f + p$ is strongly convex, the minimizer $\mathbf{x}^*$ of (5.48) is uniquely determined.

Using Lemma 5.20, we obtain

$$\begin{aligned}\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_2 &= \left\| \operatorname{prox}_{\alpha_k p}\left(\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)\right) - \operatorname{prox}_{\alpha_k p}\left(\mathbf{x}^\star - \alpha_k \nabla f(\mathbf{x}^\star)\right)\right\|_2 \\ &\leq \|\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) - \mathbf{x}^\star + \alpha_k \nabla f(\mathbf{x}^\star)\|_2.\end{aligned}$$

Using (5.50) one can show that

$$\|\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k) - \mathbf{x}^\star + \alpha_k \nabla f(\mathbf{x}^\star)\|_2 \leq \max\{|1 - \alpha_k \mu|, |1 - \alpha_k L|\}\|\mathbf{x}_k - \mathbf{x}^\star\|_2.$$

Choosing the factor $\alpha_k \equiv \frac{2}{\mu + L}$ minimizes the first factor (compare with Theorem 4.23!), which yields

$$\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\mathbf{x}_0 - \mathbf{x}^\star\|_2$$

with $\kappa = L/\mu$.

5.4.3 Quadratic programs with box constraints

It is illustrative to see how simple the projected gradient scheme becomes when applied to a quadratic program with box constraints on $\mathbf{x}$:

$$\begin{aligned}\min_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G \mathbf{x} + \mathbf{x}^T \mathbf{h} \\ \text{subject to} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\end{aligned} \tag{5.51}$$

where $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ are vectors containing lower and upper bounds on the entries of $\mathbf{x}$. We explicitly allow for entries $-\infty$ and ∞ in $\mathbf{l}$ and $\mathbf{u}$, respectively. The admissible set is clearly convex:

$$\Omega = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}.$$

Thus, (5.51) is equivalent to

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + i_\Omega(\mathbf{x}). \tag{5.52}$$

Note that the proximity operator is given by

$$\big(\operatorname{prox}_{i_\Omega}(\mathbf{y})\big)_i := \begin{cases} l_i & \text{if } y_i < l_i, \\ y_i & \text{if } y_i \in [l_i, u_i] \\ u_i & \text{if } y_i > u_i. \end{cases}$$

Since the gradient of f at a point $\mathbf{y}$ is given by $G\mathbf{y} + \mathbf{h}$, the projected gradient scheme takes the form

$$\mathbf{x}_{k+1} = \operatorname{prox}_{i_\Omega}\big(\mathbf{x}_k - \alpha_k(G\mathbf{x}_k + \mathbf{h})\big), \qquad k = 0, 1, \ldots,$$

In the special case (5.51), a good step size α_k can be determined from the so called Cauchy point $\mathbf{x}^c$, which is the first local minimizer of the piecewise quadratic function

$$t \mapsto f(\mathbf{x}(\alpha)),$$

where $x(\alpha) := \operatorname{prox}_{i_\Omega}\big(\mathbf{x}_k - \alpha(G\mathbf{x}_k + \mathbf{h})\big)$. This computation is complicated by the fact that $\mathbf{x}(\alpha)$ has kinks and hence $f(\mathbf{x}(\alpha))$ is not differentiable. We therefore need to subsequently consider subintervals on which $\mathbf{x}(\alpha)$ is smooth. Once this is done, the Cauchy point $\mathbf{x}^c$ could be readily used as the next iterate. However, it turns out to be beneficial to further optimize the non-active components of $\mathbf{x}^c$ by approximately solving the corresponding QP with a few steps of an iterative method. The technical details can be found in Section 16.7 of [NW].