Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Lecture Notes

Numerical Algorithms for Visual Computing III Summer Semester 2011

Michael Breuß
LaTeX by Kai Hagenburg
2011-7-12

Contents

1 Variational Methods I: Introduction

Motivation

Variational models play a significant role in image processing and computer vision, e.g. for

- segmentation,
- optic flow, etc.

In order to give an idea of variational problems, we focus now on a simple 1-D setting, discussing in some detail:

- how a variational problem arises
- Euler-Lagrange-equations
- convex and nonconvex problems
- constraints

1.1 A model problem

The variational problem is to find the infimum of the integrals

$$I(u) \equiv I(u(x)) \quad = \quad \int_\Omega F(x, u(x), u'(x)) \, \mathrm{d}x, \tag{1.1}$$

where $x \in \mathbb{R}$ and $\Omega = (a, b) \subset \mathbb{R}$ is an interval.

For the beginning, we assume that there is a constraint of the form

$$u(x) \quad := \quad u_0(x) \qquad \text{for } x \in \partial\Omega = \{a, b\}, \tag{1.2}$$

however, also other kinds of constraints - e.g. on $u'(x)$ - will be discussed later. The proposed task consists in finding a function $\mathcal{U}(x)$, $\mathcal{U} : (a, b) \to \mathbb{R}$, that is admissible according to the constraint imposed on competing functions $u(x)$ such that the integral

$$\int_\Omega F(x, \mathcal{U}(x), \mathcal{U}'(x)) \, \mathrm{d}x \tag{1.3}$$

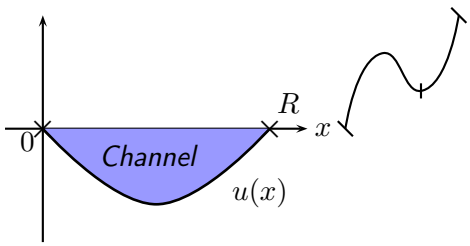
is smaller than (or equal to) the same integral for any other feasible function u.
The integrand

$$F : \Omega \times \mathbb{R} \times \mathbb{R} \to \mathbb{R} \tag{1.4}$$

is sometimes called the Lagrangian.

Let us see now how a variational optimisation problem arises.

Example 1.1 *We consider a minimal surface problem meaningful in image processing. As the corresponding image processing setting is not self-explanatory, we use here a simple equivalent physical setting for the modeling.*
We consider the profile of a channel of length L with the following geometry:



The channel shall be filled, e.g. with water, so that the amount of water W the channel can transport at any point $z \in [0, L]$ is determined by its profile:

$$W(z) \quad = \quad \int_0^R |u(x)| \, \mathrm{d}x$$

The loss of water through the boundary of the channel is proportional to the surface of the boundary, i.e., to the length of the curve $C(x) := (x, u(x))^\top \subset \mathbb{R}^2$. This length is determined as follows:

A standard formula for the length of $C(x)$, for x over $(0, R)$, is

$$\begin{aligned} \int_0^R \left\| C'(x) \right\|_2 \, \mathrm{d}x \quad &= \quad \int_0^R \left\| \begin{pmatrix} \frac{\mathrm{d}}{\mathrm{d}x}[x] \\ \frac{\mathrm{d}}{\mathrm{d}x}[u(x)] \end{pmatrix} \right\|_2 \mathrm{d}x \\ &= \quad \int_0^R \left\| \begin{pmatrix} 1 \\ u'(x) \end{pmatrix} \right\|_2 \mathrm{d}x = \int_0^R \sqrt{1 + [u'(x)]^2} \, \mathrm{d}x. \end{aligned}$$

Consider now the problem, that the channel has to transport a prescribed amount of water $\overline{W}$ at every point $z \in [0, L]$, and we aim to minimise the loss of water on the way from 0 to L, thus asking for the shape of the optimal profile $u(x)$.

The optimisation problem reads:
Minimise $\int_0^R \sqrt{1 + [u'(x)]^2} \, \mathrm{d}x$ *under the constraints:*

(i) $u(0) = 0$,

(ii) $u(R) = 0$,

(iii) $\int_0^R |u(x)| \, \mathrm{d}x = \overline{W}$.

1.2 The Euler-Lagrange equation

A usual strategy to solve the optimisation problems is to look for necessary conditions an optimal solution must satisfy. By exploiting such conditions, optimal solutions can be found or approximated explicitly in a variety of situations.

Simple example:
In attempting to find the minimum of a function $f(x)$, one can look for all points x with $f'(x) = 0$.

We explore in this paragraph the following theorem, formulated in 1-D for the cost function

$$I(u) \quad = \quad \int_\Omega F(x, u(x), u'(x)) \, \mathrm{d}x \tag{1.5}$$

For this, we use the variables λ, η within the Lagrangian F, writing formally:

$$F(x, \lambda, \eta) : \Omega \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}, \tag{1.6}$$

where F is assumed to be twice differentiable with respect to (λ, η).

Theorem 1.1 *Under the setting described above:*

1. *If u is an optimal solution, then u must also be a solution of the problem "(E-L)"*

$$\begin{cases} \dfrac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x, u(x), u'(x))] = F_\lambda(x, u(x), u'(x)) & \text{in } \Omega, \text{ and} \\ u \equiv u_0 & \text{on } \partial\Omega. \end{cases} \tag{1.7}$$

   The differential equation in (1.7) is called Euler-Lagrange equation.

2. *If u satisfies (E-L) and F is convex with respect to the variables (λ, η) for each fixed $x \in \Omega$, then u is also an optimal solution of the variational problem.*

3. *If, in addition, F is strictly convex with respect to (λ, η) for each $x \in \Omega$, the optimal solution u is unique, provided it exists.*

Remark: F depends formally on (x, λ, η), and it is just evaluated at $(x, u(x), u'(x))$. We will see in computations why this notation may help to avoid errors.

Example 1.2 *Assume the most simple situation in which F depends on η exclusively, i.e. $F = F(\eta)$. For this, (E-L) simplifies as follows*

$$\frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,u,u')] \equiv \frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(u')] = \frac{\mathrm{d}}{\mathrm{d}x}[F'(u')], \tag{1.8}$$

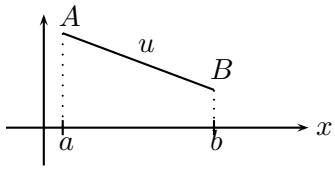
and $F_\lambda(x,u,u') \equiv F_\lambda(u') = 0$, so that we obtain

$$\frac{\mathrm{d}}{\mathrm{d}x}[F'(u')] = 0, \tag{1.9}$$

which in turn holds if

$$F'(u') = k, \qquad k \text{ constant}. \tag{1.10}$$

Evidently, this last requirement is fulfilled, if we take u' constant throughout the inverval (a,b) and this is the case if u is indeed the straight line joining the points $(a,A),(b,B)$.



Sketch:

Note:

(i) Such a linear function is always a solution of (E-L) if F only depends on η.

(ii) If, in addition, F is convex, that linear function will be a minimiser.

(iii) If, in addition, F is strictly convex, this linear function will be the only minimiser of the problem.

Example 1.3 (Non-convexity) *Let us take*

$$F \equiv F(\eta) = e^{-\eta^2}, \qquad a = A = B = 0, b = 1. \tag{1.11}$$

In this situation, the linear function through the points $(a,A) = (0,0)$ and $(b,B) = (1,0)$ is the function $\tilde{u} \equiv 0$ on the whole interval $(0,1)$. Its cost is

$$\int_0^1 F(x,\tilde{u},\tilde{u}')\,\mathrm{d}x \overset{(1.11)}{=} \int_0^1 e^{-(\tilde{u}')^2}\,\mathrm{d}x = \int_0^1 \underbrace{e^{-0^2}}_{=1}\,\mathrm{d}x = 1 \tag{1.12}$$

However let us consider the sequence of feasible functions

$$u_j(x) = j(\underbrace{x - \frac{1}{2}}_{=\frac{1}{4} \text{ for } x\in\{0,1\}=\{a,b\}})^2 - \frac{j}{4}. \tag{1.13}$$

We compute

$$u_j'(x) \;=\; 2j(x-\frac{1}{2}) = j(2x-1), \tag{1.14}$$

so that

$$\lim_{j\to\infty} e^{-(u_j'(x))^2} \;=\; \lim_{j\to\infty} e^{-j^2(2x-1)^2} = \begin{cases} e^{-\infty}\equiv 0, & x\neq\frac{1}{2} \\ e^0 = 1, & x=\frac{1}{2} \end{cases} \tag{1.15}$$

As the measure of the point $x=\frac{1}{2}$ is zero, the contribution by $e^0=1$ under the cost integral is zero, and we obtain the cost

$$\lim_{j\to\infty}\int_0^1 e^{-(u_j'(x))^2}\,\mathrm{d}x \;=\; \int_0^1 0\,\mathrm{d}x = 0, \tag{1.16}$$

i.e., the infimum of the integrals

$$\mathcal{I}(u) \;=\; \int_0^1 e^{-(u'(x))^2}\,\mathrm{d}x \tag{1.17}$$

subject to $u(0)=u(1)=0$ is zero. Hence $\tilde{u}$ is a solution of (E-L), but it is <u>not</u> a minimiser. What fails here is the <u>convexity</u> of F. It is even true that there is no minimiser for this problem, because such a minimizer $v(x)$ would have to satisfy

$$\int_0^1 e^{-(v'(x))^2}\,\mathrm{d}x \;=\; 0,$$

which is impossible for a differentiable $v(x)$.

When the integrand F depends on both x and η, (E-L) becomes

$$\frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,u')] \;=\; F_\lambda(x,u')\equiv 0, \tag{1.18}$$

or equivalently,

$$F_\eta(x,u') \;=\; \text{constant}. \tag{1.19}$$

Depending on the particular form of F, (1.19) will be solvable or not.

Example 1.4 (of Weierstrass, concerned with constraints) *Let*

$$F(x,\eta) \;=\; x\eta^2, \qquad x\in(0,1), \tag{1.20}$$

and $u(0) = 1, u(1) = 0$. *We compute (E-L):*

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,\eta)] && = F_\lambda(x,\lambda) \\ \Leftrightarrow \quad & \frac{\partial}{\partial x}\left[\frac{\partial}{\partial \eta}[x\eta^2]\right] && = \frac{\partial}{\partial \lambda}[x\eta^2] \\ \Leftrightarrow \quad & \frac{\partial}{\partial x}[2x\eta] && = 0 \\ \Leftrightarrow \quad & 2x\eta && = c' := \ \mathit{constant} \\ \Leftrightarrow \quad & x\eta && = \ \mathit{constant}, \end{aligned}$$

i.e., with $\eta \equiv u'(x)$, *we obtain the Euler-Lagrange equation*

$$x \cdot u'(x) \quad = \quad c. \tag{1.21}$$

As (1.21) *is equivalent to* $u'(x) = c \cdot \frac{1}{x}$ *(for* $x \neq 0$*) this yields*

$$u(x) \quad = \quad c \cdot \ln(x) + d, \tag{1.22}$$

where c, d *are arbitrary constants. We find:* $\ln(\varepsilon) \mapsto -\infty$ *for* $\varepsilon \downarrow 0$, *implying* $c = 0$ *and* $d = 1$ *by the condition* $u(0) = 1$. *However, since* $u(1) = 0$ *is not possible by this choice, no solution exists.*

Summary

- In variational problems, one needs to be very careful with the properties of the Lagrangian as well as with the constraints.
- Also very simple settings may impose severe difficulties.

2 Variational Methods II: The Euler-Lagrange equation

Motivation

- The Euler-Lagrange equation (E-L) constitutes many successful schemes in image processing and computer vision.
- Looking at the basics helps for a better understanding of models.

We will dwell on the issue, why optimal solutions must also be solutions of (E-L) at the level of underlying ideas. We will treat a simple 1-D case, and then we will indicate extensions.

2.1 Justification of the Euler-Lagrange equation

The 1-D Euler-Lagrange equation reads as

$$\frac{\mathrm{d}}{\mathrm{d}x}[F_{u'}(x,u,u')] \;=\; F_u(x,u,u'), \tag{2.1}$$

see (E-L) from (1.7). We do not want to omit the other ingredients of interest:

$$u(a)=A, \qquad u(b)=B, \qquad \mathcal{I}(u)=\int_a^b F(x,u(x),u'(x))\,\mathrm{d}x \tag{2.2}$$

Let φ be a fixed function satisfying the requirements $\varphi(a)=\varphi(b)=0$, and let us consider the following function of a single variable

$$\begin{aligned} g(t) &:= \mathcal{I}(u+t\cdot\varphi) \\ &= \int_a^b F(x,u(x)+t\cdot\varphi(x),u'(x)+t\cdot\varphi'(x))\,\mathrm{d}x \end{aligned} \tag{2.3}$$

Thereby, we assume that u is an optimal solution yielding the minimal value of the above integrals among all feasible functions. φ may be understood as a pertubation of u.

For each choice of $t \in \mathbb{R}$, the function $u + t \cdot \varphi$ turns out to be admissible, because φ vanishes on the endpoints of the interval (a, b). Thus, g has a minimum for $t = 0$.

A necessary condition for the occurence of a local minimum is, that the derivative vanishes at such a point:

$$\begin{aligned} 0 = \frac{\mathrm{d}}{\mathrm{d}t} g(t) &= \frac{\mathrm{d}}{\mathrm{d}t} \int_a^b F(x, u + t \cdot \varphi, u' + t \cdot \varphi') \, \mathrm{d}x && (2.4) \\ &= \int_a^b \frac{\mathrm{d}}{\mathrm{d}t} F(x, u + t \cdot \varphi, u' + t \cdot \varphi') \, \mathrm{d}x, && (2.5) \end{aligned}$$

assuming that we can perform the latter operation. We now compute the total differential below the latter integral. For this, we set:

$$\begin{cases} F \equiv F(x, \lambda, \eta), \\ x(t) = x \qquad \text{(constant with respect to } t\text{)}, \\ \lambda(t) = u + t \cdot \varphi, \\ \eta(t) = u' + t \cdot \varphi' \end{cases} \tag{2.6}$$

Then the total differential reads as:

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}t} F(x(t), \lambda(t), \eta(t)) \\ = \; & \frac{\partial F}{\partial x}(x, \lambda, \eta) \cdot \frac{\mathrm{d}x}{\mathrm{d}t}(t) + \frac{\partial F}{\partial \lambda}(x, \lambda, \eta) \cdot \frac{\mathrm{d}\lambda}{\mathrm{d}t}(t) + \frac{\partial F}{\partial \eta}(x, \lambda, \eta) \cdot \frac{\mathrm{d}\eta}{\mathrm{d}t}(t) \end{aligned} \tag{2.7}$$

With $\frac{\mathrm{d}x}{\mathrm{d}t}(t) = 0$, $\frac{\mathrm{d}\lambda}{\mathrm{d}t}(t) = \varphi$, $\frac{\mathrm{d}\eta}{\mathrm{d}t}(t) = \varphi'$ we obtain

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} F(x, \lambda, \eta) &= F_\lambda(x, \lambda, \eta) \cdot \varphi + F_\eta(x, \lambda, \eta) \cdot \varphi' && (2.8) \\ &\overset{(2.6)}{=} F_\lambda(x, u + t\varphi, u' + t\varphi')\varphi + F_\eta(x, u + t\varphi, u' + t\varphi') \cdot \varphi'. \end{aligned}$$

Plugging (2.8) into (2.5), and evaluating the integral at $t = 0$, yields:

$$\underbrace{g'(t)\Big|_{t=0} = 0}_{\text{as } u \text{ is optimal}} = \int_a^b \left[F_\lambda(x, u, u') \cdot \varphi + F_\eta(x, u, u') \cdot \varphi' \right] \mathrm{d}x \tag{2.9}$$

In order to get rid off the derivative in the term φ', we perform a partial integration

of the second part of the integral:

$$\begin{aligned}\int_a^b F_\eta(x,u,u')\varphi' \,\mathrm{d}x &= \Big[F_\eta(x,u,u')\varphi\Big]_a^b - \int_a^b \frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,u,u')]\cdot\varphi \,\mathrm{d}x \\ &= F_\eta(b,u(b),u'(b))\underbrace{\varphi(b)}_{=0} - F_\eta(a,u(a),u'(a))\cdot\underbrace{\varphi(a)}_{=0} \\ &\quad - \int_a^b \frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,u,u')]\varphi \,\mathrm{d}x \\ &= -\int_a^b \frac{\mathrm{d}}{\mathrm{d}x}[F_\eta(x,u,u')]\varphi \,\mathrm{d}x \qquad (2.10)\end{aligned}$$

Putting the result back into (2.9) yields

$$0 = \int_a^b [F_\eta(x,u,u') - \frac{\mathrm{d}}{\mathrm{d}x}F_\eta(x,u,u')]\cdot\varphi \,\mathrm{d}x \qquad (2.11)$$

Since φ is arbitrary, save for its vanishing values at a, b, the identity (2.11) can happen only if the expression within brackets vanishes identically to zero over (a, b): this is the Euler-Lagrange equation, compare Theorem 1.1. The other parts of that theorem we skip here as they require more sophisticated mathematical arguments.

2.2 Multiple dimensions

The extension to $n \geq 1$ dimensions of the 1-D (E-L), where we use again $F \equiv F(x,\lambda,\eta)$, $F : \Omega \times \mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}$ is as follows:

$$\begin{cases} \mathrm{div}(F_\eta(x,u,\nabla u)) = F_\lambda(x,u,\nabla u) & \text{in } \Omega \subset \mathbb{R}^n \\ u = u_0 & \text{on } \partial\Omega. \end{cases} \qquad (2.12)$$

Note, that η is now a vector in $\mathbb{R}^n$. The output for $F \equiv F(\nabla u)$, i.e. for $F_\eta(\nabla u)$, is a vector, e.g. for $\Omega \subset \mathbb{R}^2$ and $\vec{x} = (x_1, x_2)^\top$ we identify

$$F_\eta(\nabla u) \equiv \begin{pmatrix} a_1(\vec{x}) \\ a_2(\vec{x}) \end{pmatrix} \equiv \vec{a}. \qquad (2.13)$$

Then the divergence operation in (2.12) consists of

$$\mathrm{div}\,\vec{a} = \nabla\cdot\vec{a} = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{pmatrix} \cdot \begin{pmatrix} a_1(\vec{x}) \\ a_2(\vec{x}) \end{pmatrix} = \frac{\partial}{\partial x_1}a_1(\vec{x}) + \frac{\partial}{\partial x_2}a_2(\vec{x}) \qquad (2.14)$$

Example 2.1 (Image inpainting with linear diffusion) *We consider the 2-D case, with $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ is the unknown function, for which u_0 is given at $\partial\Omega$. In 2-D, let us denote spatial variables by x and y. We would like to minimise the integral*

$$\mathcal{I}(u) \;=\; \frac{1}{2}\int_\Omega \|\nabla u\|_2^2 \,\mathrm{d}x\,\mathrm{d}y, \tag{2.15}$$

where

$$\|\nabla u\|_2^2 \;=\; \left\| \begin{pmatrix} u_x \\ u_y \end{pmatrix} \right\|_2^2 = \left(\sqrt{u_x^2+u_y^2}\right)^2 = u_x^2+u_y^2. \tag{2.16}$$

We notice that the Lagrangian

$$F(\eta) \;=\; \frac{1}{2}\|\eta\|_2^2 = \frac{1}{2}|\eta|^2. \tag{2.17}$$

is a strictly convex function of η. For the computation of $F_\eta(x,\lambda,\eta)$, we basically treat η as if it is a scalar number, although it is a vector. Thus, we obtain here

$$F_\eta(\eta) \;\equiv\; F'(\eta) = \left[\frac{1}{2}|\eta|^2\right]' = \left[\frac{1}{2}\eta^2\right] = \eta. \tag{2.18}$$

Then again, for the evaluation of

$$\mathrm{div}\,\left(F'(\eta)\big|_{\eta=\nabla u}\right), \tag{2.19}$$

we need that ∇u is a vector:

$$F'(\eta)\big|_{\eta=\nabla u} \overset{(2.18)}{=} \eta\big|_{\eta=\nabla u} = \nabla u, \tag{2.20}$$

so that

$$\begin{aligned} \mathrm{div}\,(F'(\nabla u)) &= \mathrm{div}\,(\nabla u) \\ &= \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} u_x \\ u_y \end{pmatrix} = \frac{\partial}{\partial x}\underbrace{u_x}_{\frac{\partial}{\partial x}u} + \frac{\partial}{\partial y}\underbrace{u_y}_{\frac{\partial}{\partial y}u} \\ &= \frac{\partial^2}{\partial x^2}u + \frac{\partial^2}{\partial y^2}u = u_{xx}+u_{yy} =: \Delta u, \end{aligned} \tag{2.21}$$

where $\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is called the Laplace-Operator. As $F \equiv F(\eta)$, we also obtain $F_\lambda(\eta) = 0$, so that the Euler-Lagrange equation finally reads as

$$\Delta u \;=\; 0 \tag{2.22}$$

also called Laplace equation, or linear diffusion equation. The corresponding necessary condition for the unique minimiser of (2.15),

$$\begin{cases} \Delta u = 0 & \text{in } \Omega \\ u = u_0 & \text{on } \partial\Omega, \end{cases} \tag{2.23}$$

is often interpreted as a stable equilibrium with respect to an energy proportional to the square of the gradient of u.

2.3 Other boundary conditions

If we do not assume $\varphi(a) = \varphi(b) = 0$ - which we have done for fixing $u(a) = A$, $u(b) = B$ - we need to make sure in the above derivation, that

$$F_\eta(b, u(b), u'(b))\varphi(b) - F_\eta(a, u(a), u'(a))\varphi(a) \quad = \quad 0. \tag{2.24}$$

For arbitrary φ, this can only be assured by

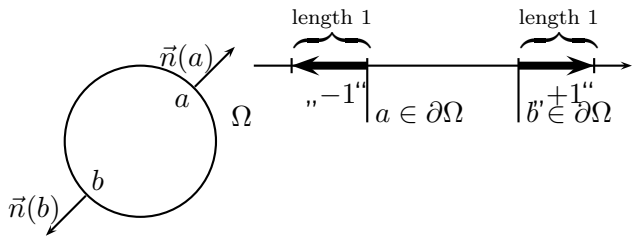
$$F_\eta(x, u, u') \quad = \quad 0 \qquad \text{at } \partial\Omega. \tag{2.25}$$

In multiple dimensions, e.g. in 2-D, (2.13) generalizes to

$$\vec{n}^\top \cdot F_\eta(x, y, \lambda, \eta)\Big|_{(x,y,u,\nabla u)} = 0. \tag{2.26}$$

The constraint (2.26) is called natural boundary conditions, $\vec{n}$ is a unit normal vector on $\partial\Omega$ (pointing outside).

Sketch:



$\|\vec{n}\|_2 = 1$

Summary

- The Euler-Lagrange equation is shown to be a necessary condition for a minimiser.
- Several settings, e.g. with multiple dimensions or different boundary constraints can be adressed by it.

3 Variational Methods III: Complex Computations

Motivation

Variational models that combine several strategies into one functional may lead to the minimisation with respect to several unknown functions at the same time.

In this paragraph, we elaborate on a simple example.

3.1 Systems of Euler-Lagrange equations

We consider the minimisation of the functional

$$E(u,v) \;=\; \int\limits_{\Omega} F(x,y,u,v,\nabla u,\nabla v)\,\mathrm{d}x\,\mathrm{d}y, \tag{3.1}$$

where $\Omega \subset \mathbb{R}^2$ and

$$u := u(x,y), \quad v := v(x,y), \quad \nabla := \begin{pmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \end{pmatrix}. \tag{3.2}$$

In order to make use of the already developed proceeding in §2.1, we also fix values of u, v at the image boundary $\partial\Omega$.

Let $\varphi := \varphi(x,y)$ and $\Psi := \Psi(x,y)$ be some fixed perturbation function with $\varphi \equiv 0, \Psi \equiv 0$ on $\partial\Omega$, and let us consider the function

$$\begin{aligned} g(t,s) \;&:=\; E(u + t\cdot\varphi, v + s\cdot\Psi) \\ &=\; \int\limits_{\Omega} F(x,y,u + t\cdot\varphi, v + s\cdot\Psi, \nabla u + t\cdot\nabla\varphi, \nabla v + s\cdot\nabla\Psi)\,\mathrm{d}x\,\mathrm{d}y \end{aligned} \tag{3.3}$$

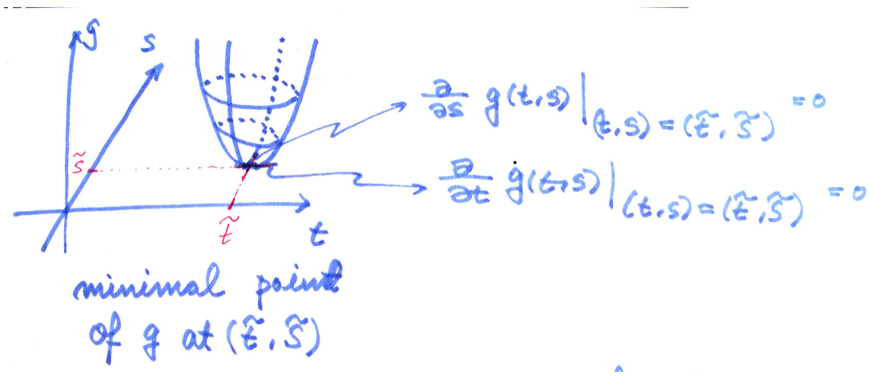
Analogously to the procedure in §2.1, we assume that u, v denote optimal solutions, and this implies that $g(t,s)$ has a local minimum at $(t,s) = (0,0)$. A necessary

condition for having a minimum of $g(t,s)$ is

$$\nabla_{(t,s)} g(t,s) \quad = \quad \begin{pmatrix} \frac{\partial}{\partial t} g(t,s) \\ \frac{\partial}{\partial s} g(t,s) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{3.4}$$

see Sketch.

Sketch:



Thus, by (3.4) we get the system of conditions

$$\begin{cases} 0 \quad = \dfrac{\partial}{\partial t} g(t,s) = \displaystyle\int_\Omega \frac{\partial}{\partial t} F(x,y,u+t\cdot\varphi, v+s\cdot\Psi, \nabla u + t\cdot\nabla\varphi, \nabla v + s\cdot\nabla\Psi)\,\mathrm{d}x\,\mathrm{d}y \\ 0 \quad = \dfrac{\partial}{\partial s} g(t,s) = \displaystyle\int_\Omega \frac{\partial}{\partial s} F(x,y,u+t\cdot\varphi, v+s\cdot\Psi, \nabla u + t\cdot\nabla\varphi, \nabla v + s\cdot\nabla\Psi)\,\mathrm{d}x\,\mathrm{d}y \end{cases} \tag{3.5}$$

compare (2.4), (2.5).

We now exercise the first differential in (3.5). For this, we set

$$F \quad \equiv \quad F(\tilde{x}, \tilde{y}, \lambda, \gamma, \eta_1, \eta_2, \xi_1, \xi_2) \tag{3.6}$$

with

$$\begin{cases} \tilde{x}(t,s) & = x \\ \tilde{y}(t,s) & = y \\ \lambda(t,s) & = u + t\cdot\varphi \\ \gamma(t,s) & = v + s\cdot\Psi \\ \eta_1(t,s) & = u_x + t\cdot\varphi_x \\ \eta_2(t,s) & = u_y + t\cdot\varphi_y \\ \xi_1(t,s) & = v_x + s\cdot\Psi_x \\ \xi_2(t,s) & = v_y + s\cdot\Psi_y \end{cases} \tag{3.7}$$

Then we obtain

$$\begin{aligned}\frac{\partial}{\partial t}F &= \frac{\partial F}{\partial \tilde{x}}\cdot\frac{\partial \tilde{x}}{\partial t}+\frac{\partial F}{\partial \tilde{y}}\cdot\frac{\partial \tilde{y}}{\partial t}+\frac{\partial F}{\partial \lambda}\cdot\frac{\partial \lambda}{\partial t}\\ &\quad+\frac{\partial F}{\partial \gamma}\cdot\frac{\partial \gamma}{\partial t}+\frac{\partial F}{\partial \eta_1}\cdot\frac{\partial \eta_1}{\partial t}+\frac{\partial F}{\partial \eta_2}\cdot\frac{\partial \eta_2}{\partial t}\\ &\quad+\frac{\partial F}{\partial \xi_1}\cdot\frac{\partial \xi_1}{\partial t}+\frac{\partial F}{\partial \xi_2}\cdot\frac{\partial \xi_2}{\partial t},\end{aligned} \tag{3.8}$$

neglecting the arguments of F and of the functions from (3.7), respectively. By

$$\frac{\partial \tilde{x}}{\partial t}=\frac{\partial \tilde{y}}{\partial t}=\frac{\partial \gamma}{\partial t}=\frac{\partial \xi_1}{\partial t}=\frac{\partial \xi_2}{\partial t}=0,\ \frac{\partial \lambda}{\partial t}=\varphi,\ \frac{\partial \eta_1}{\partial t}=\varphi_x,\ \frac{\partial \eta_2}{\partial t}=\varphi_y, \tag{3.9}$$

we get in total:

$$\frac{\partial}{\partial s}F=\frac{\partial F}{\partial \lambda}\cdot\varphi+\frac{\partial F}{\partial \eta_1}\cdot\varphi_x+\frac{\partial F}{\partial \eta_2}\cdot\varphi_y. \tag{3.10}$$

Analogously, we can compute

$$\frac{\partial}{\partial s}F=\frac{\partial F}{\partial \gamma}\cdot\Psi+\frac{\partial F}{\partial \xi_1}\cdot\Psi_x+\frac{\partial F}{\partial \xi_2}\cdot\Psi_y. \tag{3.11}$$

Let us remark here that one could have simplified these computations slightly making use of $F=F(\tilde{x},\tilde{y},\lambda,\gamma,\eta,\xi)$ with vector-valued variables

$$\eta(t,s)=\nabla u+t\cdot\nabla\varphi,\qquad \xi(t,s)=\nabla v+s\cdot\nabla\Psi. \tag{3.12}$$

In the stages of the above computation, this would have lead to

$$\begin{aligned}\frac{\partial}{\partial t}F &= \frac{\partial F}{\partial \tilde{x}}\cdot\frac{\partial \tilde{x}}{\partial t}+\frac{\partial F}{\partial \tilde{y}}\cdot\frac{\partial \tilde{y}}{\partial t}+\frac{\partial F}{\partial \lambda}\cdot\frac{\partial \lambda}{\partial t}+\frac{\partial F}{\partial \gamma}\cdot\frac{\partial \gamma}{\partial t}\\ &\quad+\frac{\partial F}{\partial \eta}\cdot\frac{\partial \eta}{\partial t}+\frac{\partial F}{\partial \xi}\cdot\frac{\partial \xi}{\partial t},\end{aligned} \tag{3.13}$$

compare (3.8), and consequently to the set of equations

$$\begin{cases}\dfrac{\partial}{\partial t}F &= \dfrac{\partial F}{\partial \lambda}\cdot\varphi+\dfrac{\partial F}{\partial \eta}\cdot\nabla\varphi,\\ \dfrac{\partial}{\partial s}F &= \dfrac{\partial F}{\partial \gamma}\cdot\Psi+\dfrac{\partial F}{\partial \xi}\cdot\nabla\Psi,\end{cases} \tag{3.14}$$

compare (3.10), (3.11). The only thing to note is that one needs to take care of the vector dimensions, as

$$\frac{\partial F}{\partial \eta}\cdot\nabla\varphi = \begin{pmatrix}\dfrac{\partial}{\partial \eta_1}F\\ \dfrac{\partial}{\partial \eta_2}F\end{pmatrix}\cdot\begin{pmatrix}\varphi_x\\ \varphi_y\end{pmatrix}=\frac{\partial F}{\partial \eta_1}\varphi_x+\frac{\partial F}{\partial \eta_2}\varphi_y, \tag{3.15}$$

$\frac{\partial F}{\partial \xi} \cdot \nabla \Psi$ being formed analogously.

Making use of (3.10), (3.11), and evaluating the integrals (3.5) at $(t, s) = (0, 0)$ gives the necessary optimality conditions

$$\begin{cases} 0 &= \int\limits_{\Omega} F_\lambda \cdot \varphi + F_{\eta_1} \cdot \varphi_x + F_{\eta_2} \cdot \varphi_y \, \mathrm{d}x \, \mathrm{d}y \\ 0 &= \int\limits_{\Omega} F_\gamma \cdot \Psi + F_{\xi_1} \cdot \Psi_x + F_{\xi_2} \cdot \Psi_y \, \mathrm{d}x \, \mathrm{d}y \end{cases} \tag{3.16}$$

Corresponding to the procedure in (2.10), we can perform integration by parts with respect to both the variables x and y. Employing also along the boundary $\varphi \equiv 0 \equiv \Psi$ we obtain

$$\begin{cases} 0 &= \int\limits_{\Omega} [F_\lambda - \frac{\partial}{\partial x} F_{\eta_1} - \frac{\partial}{\partial y} F_{\eta_2}] \cdot \varphi \, \mathrm{d}x \, \mathrm{d}y \\ 0 &= \int\limits_{\Omega} [F_\gamma - \frac{\partial}{\partial x} F_{\xi_1} - \frac{\partial}{\partial y} F_{\xi_2}] \cdot \Psi \, \mathrm{d}x \, \mathrm{d}y, \end{cases} \tag{3.17}$$

compare (2.11). With the more compact notation of divergence from (2.14) we thus obtain the system of Euler-Lagrange equations:

$$\begin{cases} 0 &= F_\lambda - \operatorname{div} F_\eta \\ 0 &= F_\gamma - \operatorname{div} F_\xi \end{cases} \tag{3.18}$$

Summary

- Minimising a functional with respect to n unknown functions gives n Euler-Lagrange equations.
- Employing a compact notation means to take care of vector dimensions.

3.2 Example from image segmentation

In 1992, Ambrosio and Tortorelli suggested the following model for image segmentation:

$$\begin{aligned} E_{AT}(u, v) \quad := \quad & \int\limits_{\Omega} \beta(u - f)^2 + v^2 |\nabla u|^2 \\ & + \alpha \cdot \left(c|\nabla v|^2 + \frac{(1 - v)^2}{4c} \right) \mathrm{d}x \, \mathrm{d}y, \end{aligned} \tag{3.19}$$

where

- u is a smoothed version of the input image f,

- v is a smooth edge detector function with $v \approx 0$ at edges and $v \approx 1$ within a region,
- α, β, c are real, user-defined parameters.

Employing $F \equiv F(\tilde{x}, \tilde{y}, \lambda, \gamma, \eta_1, \eta_2, \xi_1, \xi_2)$ with

$$\begin{cases} \tilde{x}(t,s) := x, & \tilde{y}(t,s) := y \\ \lambda := \lambda(u) = u & \\ \gamma := \gamma(v) = v & \\ \eta_1 := \eta_1(u) = u_x & \eta_2 := \eta_2(u) = u_y \\ \xi_1 := \xi_1(v) = v_x & \xi_2 := \xi_2(v) = v_y \end{cases} \tag{3.20}$$

$$F \quad = \quad \beta(\lambda - f)^2 + \gamma^2(\eta_1^2 + \eta_2^2) + \alpha \cdot (c(\xi_1^2 + \xi_1^2) + \frac{(1-\gamma)^2}{4c})$$

gives

- by

$$\begin{cases} F_\lambda & = 2\beta(\lambda - f) \\ F_{\eta_1} & = 2\gamma^2\eta_1 \\ F_{\eta_2} & = 2\gamma^2\eta_2 \end{cases} \tag{3.21}$$

  and by evaluating the terms at (u, v) the first Euler-Lagrange equation:

$$0 \quad = \quad \beta(u - f) - \text{div } (\gamma^2 \nabla u) \tag{3.22}$$

- by

$$\begin{cases} F_\gamma & = 2\gamma(\eta_1^2 + \eta_2^2) - \dfrac{2\alpha}{4c}(1-\gamma) \\ F_{\xi_1} & = 2\alpha c \xi_1 \\ F_{\xi_2} & = 2\alpha c \xi_2 \end{cases} \tag{3.23}$$

  and by evaluating the terms at (u, v) the second Euler-Lagrange equation:

$$0 \quad = \quad v|\nabla u|^2 - \frac{\alpha}{4c}(1-v) - \text{div } (\alpha c \nabla v). \tag{3.24}$$

4 Numerical Treatment of Variational Problems

Motivation

How do we implement variational models on a computer?

We discuss in detail the 1-D case, the extension to higher dimensions is straightforward. We employ the model problem

$$E(u) \quad = \quad \int_a^b (u-f)^2 + \alpha\Psi((u')^2)\,\mathrm{d}x \tag{4.1}$$

with natural boundary constraints, which we assume here to lead to homogeneous Neumann boundary conditions:

$$u'(x) \quad = \quad 0 \qquad \text{for } x = a, b. \tag{4.2}$$

Computing the Euler-Lagrange equation for (4.1), we have the ingredients:

$$\begin{cases} F(x,u,u') & = (u-f)^2 + \alpha\Psi((u')^2) \\ F_u(x,u,u') & = 2(u-f) \\ F_{u'}(x,u,u') & = \alpha\Psi'((u')^2)\cdot 2u' \end{cases} \tag{4.3}$$

so that we obtain

$$2(u-f) \quad = \quad \frac{\mathrm{d}}{\mathrm{d}x}[2\alpha\Psi'((u')^2)u'], \tag{4.4}$$

i.e. as α is a scalar parameter independent of x,

$$\frac{u-f}{\alpha} \quad = \quad \frac{\mathrm{d}}{\mathrm{d}x}[\Psi'((u')^2)u']. \tag{4.5}$$

We now discuss how to solve (4.5).

4.1 Discretisation of the Euler-Lagrange equation

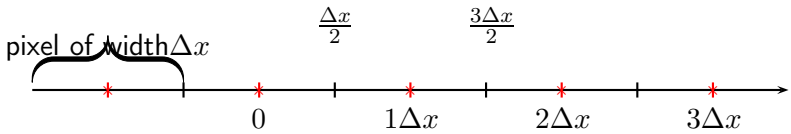
We may discretise all occurring derivatives in (4.5) by use of pixel-wise given values

$$u_j \quad \equiv \quad u(x_j), \tag{4.6}$$

where x_j is the center of the j-th pixel. The pixel width is given by the spatial mesh parameter Δx:

$$u_j \quad \equiv \quad u(x_j) = u(j\Delta x) \tag{4.7}$$

Sketch pixel boundaries at $(j \pm \frac{1}{2})\Delta x$ for pixel number j



Remarks:

(i) One can also center pixels at $(j \pm \frac{1}{2})\Delta x$, and use pixel boundaries at $j\Delta x$. This is sometimes useful.

(ii) In image processing, often $\Delta x = 1$ is chosen. However, we recommend to write all formulae with Δx and not with that special choice, as this proceeding enables to avoid many errors.

Popular and simple finite differences are:

$$\begin{cases} u_j' \approx \dfrac{u_{j+1} - u_j}{\Delta x} & \text{forward difference, first-order discretisation,} \\ u_j' \approx \dfrac{u_j - u_{j-1}}{\Delta x} & \text{backward difference, first-order discretisation,} \\ u_j' \approx \dfrac{u_{j+1} - u_{j-1}}{2\Delta x} & \text{central difference, second-order discretisation.} \end{cases} \tag{4.8}$$

Higher order derivative discretisations, e.g. u_j'', can be computed by concatenating first order derivative discretisations from (4.8).

Example 4.1 *Rewriting* (4.5)*, we obtain pixelwise*

$$0 \quad = \quad u_j - f_j - \alpha \frac{\mathrm{d}}{\mathrm{d}x} \left[\Psi'((u')^2) u' \right] \big|_{x=x_j} \tag{4.9}$$

Now, we employ:

- $u'(x_j) \approx \dfrac{u_j - u_{j-1}}{\Delta x}$ *(backward difference)*
- $\dfrac{\mathrm{d}}{\mathrm{d}x}[\ldots]\Big|_{x=x_j} \approx \dfrac{[\ldots]|_{x_{j+1}} - [\ldots]|_{x_j}}{\Delta x}$ *(forward difference).*

This gives

$$\begin{aligned}
& \frac{\mathrm{d}}{\mathrm{d}x}[\Psi'((u')^2)u']\Big|_{x=x_j} \\
\approx\ & \frac{[\Psi'([u'(x_{j+1})]^2)u'(x_{j+1})] - [\Psi'([u'(x_j)]^2)u'(x_j)]}{\Delta x} \\
\approx\ & \frac{1}{\Delta x}\left(\Psi'\left(\left[\frac{u_{j+1}-u_j}{\Delta x}\right]^2\right)\frac{u_{j+1}-u_j}{\Delta x} - \Psi'\left(\left[\frac{u_j-u_{j-1}}{\Delta x}\right]^2\right)\frac{u_j-u_{j-1}}{\Delta x}\right) \\
=\ & \Psi'\left(\frac{(u_{j+1}-u_j)^2}{\Delta x^2}\right)\frac{u_{j+1}-u_j}{\Delta x^2} - \Psi'\left(\frac{(u_j-u_{j-1})^2}{\Delta x^2}\right)\frac{u_j-u_{j-1}}{\Delta x^2} \qquad (4.10)
\end{aligned}$$

For a grid with pixels $j = 1, \ldots, N$, the boundary conditions (4.2) can be realised via two dummy pixels with numbers 0 and $N+1$, and corresponding ghost data

$$u_0 := u_1, \qquad u_{N+1} := u_N. \qquad (4.11)$$

Note that it also depends on discretisation: the central difference leads to $u_0 := u_2$ and $u_{N+1} := u_{N-1}$.

The mathematical task boils down to solving the (nonlinear) system of equations

$$\begin{aligned}
0 \ =\ & u_j - f_j - \alpha\Psi'\left(\frac{(u_{j+1}-u_j)^2}{\Delta x^2}\right)\frac{u_{j+1}-u_j}{\Delta x^2} \qquad (4.12) \\
& +\alpha\Psi'\left(\frac{(u_j-u_{j-1})^2}{\Delta x^2}\right)\frac{u_j-u_{j-1}}{\Delta x^2}, \qquad j = 1, \ldots, N.
\end{aligned}$$

Remarks:

(i) We assume here, that the derivative of Ψ can be computed analytically.

(ii) One may employ different choices than those in this example. The effect of the above choice is, that, in (4.12) the j-th equation only depends on $\{u_{j-1}, u_j, u_{j+1}\}$.

Example 4.2 *Writing again*

$$\frac{u-f}{\alpha} \ =\ \frac{\mathrm{d}}{\mathrm{d}x}[\Psi'((u')^2)u'], \qquad (4.13)$$

we may understand α as an artificial time variable, by which we obtain for $\alpha \searrow 0$

$$\frac{\partial}{\partial t}u(x,t) \ =\ \frac{\partial}{\partial x}\left[\Psi'\left(\left[\frac{\partial}{\partial x}u(x,t)\right]^2\right)\frac{\partial}{\partial x}u(x,t)\right]. \qquad (4.14)$$

In (4.14),

- we have augmented the unknown function u by an argument depending on the introduced artificial time:

$$u(x) \mapsto u(x,t),$$

- the spatial derivative in u' goes over to a partial derivative depending on x:

$$u'(x) \mapsto \frac{\partial}{\partial x} u(x,t).$$

The partial differential equation (PDE) (4.14) needs to be solved by use of:

(i) the initial condition

$$u(x,0) \quad = \quad f(x) \tag{4.15}$$

and

(ii) at any time t with Neumann boundary conditions

$$\frac{\partial}{\partial x} u(x,t) \quad = \quad 0\big|_{x\in\{a,b\}} \tag{4.16}$$

Technically, one may understand this way to solve (4.5) as a parametrisation of α, where $t \in [0,\alpha]$. The „stopping time" when integrating (4.5) is thus the parameter α. A simple and popular way to deal with $\frac{\partial}{\partial t}$ is to use the Euler forward descretisation:

$$t \quad \equiv \quad n \cdot \Delta t =: t^n \qquad \frac{\partial}{\partial t} u(x,t^n) \approx \frac{u(x,(n+1)\Delta t) - u(x,n\Delta t)}{\Delta t}, \tag{4.17}$$

where n is to be understood as time step number and Δt as the time size.

This explicit time stepping choice, i.e. all terms with derivatives are considered to be given, yields very simple iterative schemes: Using this together with formula (4.10), we obtain

$$\begin{aligned} \frac{u_j^{n+1} - u_j^n}{\Delta t} \quad = \quad & \Psi'\left(\frac{(u_{j+1}^n - u_j^n)^2}{\Delta x^2}\right) \frac{u_{j+1}^n - u_j^n}{\Delta x^2} \\ & -\Psi'\left(\frac{(u_j^n - u_{j-1}^n)^2}{\Delta x^2}\right) \frac{u_j^n - u_{j-1}^n}{\Delta x^2} \end{aligned} \tag{4.18}$$

where the upper index denotes the time level, and $u^0 := f_j$.

5 Numerical Treatment of Variational Problems II: Direct Optimisation Approach

Motivation

Is it possible to deal with variational models without resorting to the Euler-Lagrange equation?

We start again from the model problem of §4 with the "continuous-scale" functional

$$E(u) \quad = \quad \int_a^b (u-f)^2 + \alpha\Psi([u']^2)\,\mathrm{d}x. \tag{5.1}$$

5.1 Integration of the Energy functional

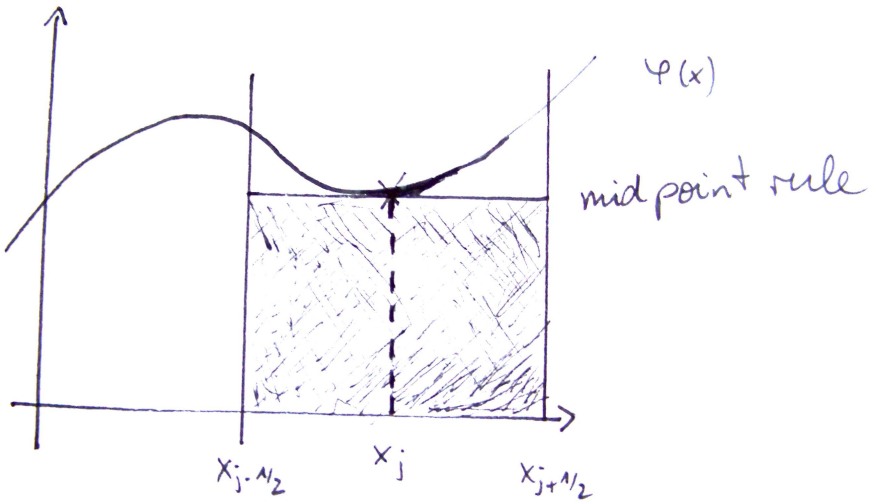
Via local quadrature (mid-point rule)

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \varphi(x)\,\mathrm{d}x \quad \approx \quad \underbrace{(x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}})}_{\Delta x}\,\varphi(x_j) \tag{5.2}$$

we obtain its discrete version

$$E_\Delta(u) \quad := \quad \Delta x \sum_{j=1}^{N} (u_j - f_j)^2 + \alpha\Psi([u']^2)\Big|_{x=x_j} . \tag{5.3}$$

Sketch



Also here, we need to discretise $u'(x)$, which we do in (5.3) by use of forward differences:

$$E_\Delta(u) \quad \approx \quad \Delta x \sum_{j=1}^{N} (u_j - f_j)^2 + \alpha \Psi \left(\frac{(u_{j+1} - u_j)^2}{\Delta x^2} \right). \tag{5.4}$$

As we seek a minimiser of $E_\Delta(u)$, where $u = (u_1, \ldots, u_N)^\top$, we can explore a necessary condition for optimality in the form of vanishing partial derivatives:

$$\frac{\partial E_\Delta(u)}{\partial u_j} \quad \stackrel{!}{=} \quad 0, \qquad \text{for } j = 1, \ldots, N. \tag{5.5}$$

Computing these partial derivatives, we obtain:

$$\begin{aligned} j = 1: \quad & 2\Delta x(u_1 - f_1) - 2\alpha \Delta x \Psi' \left(\frac{(u_2 - u_1)^2}{\Delta x^2} \right) \frac{u_2 - u_1}{\Delta x^2} \\ & + 2\alpha \Delta x \Psi' \left(\frac{(u_1 - u_0)^2}{\Delta x^2} \right) \frac{u_1 - u_0}{\Delta x^2} \stackrel{!}{=} 0 \end{aligned} \tag{5.6a}$$

$$\begin{aligned} j = 2, \ldots, N-1: \quad & 2\Delta x(u_j - f_j) - 2\alpha \Delta x \Psi' \left(\frac{(u_{j+1} - u_j)^2}{\Delta x^2} \right) \frac{u_{j+1} - u_j}{\Delta x^2} \\ & + 2\alpha \Delta x \Psi' \left(\frac{(u_j - u_{j-1})^2}{\Delta x^2} \right) \frac{u_j - u_{j-1}}{\Delta x^2} \stackrel{!}{=} 0 \end{aligned} \tag{5.6b}$$

$$\begin{aligned} j = N: \quad & 2\Delta x(u_N - f_N) - 2\alpha \Delta x \Psi' \left(\frac{(u_{N+1} - u_N)^2}{\Delta x^2} \right) \frac{u_{N+1} - u_N}{\Delta x^2} \\ & + 2\alpha \Delta x \Psi' \left(\frac{(u_N - u_{N-1})^2}{\Delta x^2} \right) \frac{u_N - u_{N-1}}{\Delta x^2} \stackrel{!}{=} 0 \end{aligned} \tag{5.6c}$$

For the boundary conditions, we set $u_0 := u_1$, $u_{N+1} := u_N$, and obtain:

$$\text{left boundary:} \qquad u_0 := u_1 \tag{5.7}$$

$$\Rightarrow \quad 2\alpha\Delta x\Psi'\left(\frac{(u_1-u_0)^2}{\Delta x^2}\right)\frac{u_1-u_0}{\Delta x^2} = 0 \qquad (\text{in (5.6a)})$$

$$\text{right boundary:} \qquad u_{N+1} := u_N \tag{5.8}$$

$$\Rightarrow \quad 2\alpha\Delta x\Psi'\left(\frac{(u_{N+1}-u_N)^2}{\Delta x^2}\right)\frac{u_{N+1}-u_N}{\Delta x^2} = 0 \qquad (\text{in (5.6c)})$$

Comparing with §4, we observe that this gives finally the same nonlinear system of equations as (4.18), but without using the Euler-Lagrange equation. Note that we have employed special choices for discretisation that "fit"!

Two questions and answers

1: Is it a surprise that we obtain by both approaches the same system of equations ?

Mathematically not, as we have explored (i) via the Euler-Lagrange equation and (ii) the condition (5.5) in both cases necessary conditions for an extremum.

2: Is there any difference at all between the two approaches, as they lead to exactly the same system to solve?

- For the Euler-Lagrange approach in §4, we first employed the necessary optimality condition, and then we discretised.
- For the direct approach above using the variational form, we first discretised, and then we used the necessary optimality condition.

5.2 Building the Bridge between Euler-Lagrange and Direct Optimisation

In order to solve the nonlinear system (5.6a)for the state $\nabla_u E_\Delta(u) = 0$ where

$$\nabla_u E_\Delta(u) \quad = \quad \left(\frac{\partial}{\partial u_1}E_\Delta(u), \ldots, \frac{\partial}{\partial u_N}E_\Delta(u)\right)^\top, \tag{5.9}$$

we may employ an iterative method such as

$$u^{k+1} \quad = \quad u_k - \gamma_k \nabla_u E_\Delta(u), \tag{5.10}$$

where γ_k is a parameter, and where we aim for a stationary case $u^{k+1} = u^k$ for all k large enough. Can we derive expressions corresponding to such an iterative method more directly by the energy functional?
Dealing with

$$E(u) \quad = \quad \int_\Omega F(\vec{x}, u(\vec{x}), \nabla u(\vec{x}))\, \mathrm{d}\vec{x}, \tag{5.11}$$

we need to find a gradient of a functional, i.e. we aim to find a derivative with respect to elements u of an infinite dimensional function space, and not with respect to a vector $(u_1, \dots, u_n)^\top$ with a finite number of entries.

Defining $\nabla_u E$:

From vector calculus, i.e., if u was a vector $\vec{u} = (u_1, \dots, u_N)^\top$, we know that the directional derivative in a direction $\vec{\varphi}$ is

$$\frac{\partial E}{\partial \varphi} = \lim_{\varepsilon \to 0} \frac{E(\vec{u} + \varepsilon\vec{\varphi}) - E(\vec{u})}{\varepsilon} = \nabla_{\vec{u}} E \cdot \vec{\varphi}, \tag{5.12}$$

where the scalar product expands as

$$\nabla_{\vec{u}} E \cdot \vec{\varphi} = \sum_{i=1}^{N} (\nabla_{\vec{u}} E)_i \varphi_i. \tag{5.13}$$

Using the same methodology, we use now functions $u(\vec{x})$ and $\varphi(\vec{x})$ instead of $\vec{u}$ and $\vec{\varphi}$, respectively, and the scalar product

$$\nabla_u E \cdot \varphi = \int_\Omega (\nabla_u E)(\vec{x}) \varphi(\vec{x}) \, \mathrm{d}\vec{x}, \tag{5.14}$$

compare (5.13). Following then the derivation of the Euler-Lagrange equation in §2, identifying

$$\frac{\mathrm{d}}{\mathrm{d}t} g(t) = \frac{\mathrm{d}}{\mathrm{d}t} E(u + t\varphi) = \lim_{t \to 0} \frac{E(u + t\varphi) - E(u)}{t} =: \nabla_u E, \tag{5.15}$$

we arrive at

$$\nabla_u E = \frac{\partial F}{\partial u} - \mathrm{div}(F_{\nabla_u}(\vec{x}, u, \nabla u)). \tag{5.16}$$

An iterative method can then modeled via the concept of artificial time,

$$\frac{\partial u}{\partial t} = -\nabla_u E, \tag{5.17}$$

i.e., after discretisation of $\frac{\partial u}{\partial t}$ by $\frac{\partial u}{\partial t} = \frac{u_j^{n+1} - u_j^n}{\Delta t}$ we obtain

$$u_j^{n+1} = u_j^n - \Delta t (\nabla_u E)\Big|_{(x_j, t^n)}. \tag{5.18}$$

Summary

We now have at hand the approaches to deal with variational optimisation problems:

- the Euler-Lagrange equation plus its discretisation
- quadrature of the energy functional
- optimisation by iterative schemes

6 Descent Methods: An Introduction

Motivation

Descent schemes are widely used in areas related to visual computing tasks, e.g. in machine learning for pattern recognition.

We begin with employing a broad perspective.

Descent methods are often employed for unconstrained minimisation problems. We cast this in the form:

$$\text{minimise } f(x) \tag{6.1}$$

where we will often assume:

$$\begin{cases} f : \mathbb{R}^n \to \mathbb{R} \\ f \text{ is convex} \\ f \text{ is twice continuously differentiable} \\ \text{the problem is solvable} \end{cases} \tag{6.2}$$

Remark: In variational formulations, we have $E(\vec{u})$ instead of $f(x)$.
We will also discuss deviations from this setting. Since f is differentiable and convex, a necessary and sufficient condition for a point x^* to be optimal is

$$\nabla f(x^*) \quad = \quad \vec{0}. \tag{6.3}$$

Thus, solving (6.1) is the same as solving (6.3), where the latter is a system of n equations in the n variables $x_1, \ldots, x_n$.

We seek algorithms constructing a minimising sequence $x^{(0)}, x^{(1)}, \ldots$, with

$$f(x^{(k)}) \quad \to \quad f(x^*) =: p^* \qquad \text{for } k \to \infty. \tag{6.4}$$

The algorithm is terminated when

$$f(x^{(k)}) - p^* \quad \leq \quad \varepsilon, \qquad \varepsilon > 0 \tag{6.5}$$

being a prescribed tolerance.

Remark: We indirectly assume above that all generated points $x^{(k)}$ are allowed.

In general it is not assumed that $f(x^{(k+1)}) < f(x^{(k)})$ in a minimising sequence, only the sequence shall converge to the minimiser, see (6.4). In a descent scheme, this is different!

6.1 Generic Algorithm Formulation

The algorithms we describe shall produce a minimising sequence $x^{(k)}$ where

$$x^{(k+1)} \quad = \quad x^{(k)} + \gamma_k d^{(k)}, \tag{6.6}$$

and where $\gamma_k > 0$, except when $x^{(k)}$ is optimal. In (6.6), we denote

- $d^{(k)} \in \mathbb{R}^n$ as the search direction
- $\gamma_k \in \mathbb{R}$ as the step size, or step length.

In a descent method we have

$$f(x^{(k+1)}) \quad < \quad f(x^{(k)}), \tag{6.7}$$

except when $x^{(k)}$ is optimal.

A general descent method alternates between two steps: determining a descent direction d, and the selection of a step size γ.

Algorithm 6.1 (General descent method)

- *give a starting point* x
- *repeat*
  - a) *Determine descent direction* d
  - b) *Line search. Choose step size* γ
  - c) *Update.* $x := x + \gamma d$

  until stopping criterion is satisfied.

The second step is called 'line search' since selection of the step size γ determines, where along the onedimensional half-line $\{x + \gamma d \mid \gamma \in \mathbb{R}_+\}$ the next iterate will be.

6.2 Basic Line Searching Strategies

There are two principle possibilities:

(i) Solve the 1-D line search problem exact.

(ii) Solve it inexact.

An exact solver is sometimes adequate, if the computational cost of the 1-D minimisation problem to minimise f along the $\{x + \gamma d \mid \gamma \geq 0\}$,

$$\gamma \quad = \quad \underset{s \geq 0}{\operatorname{argmin}} \; f(x + sd), \tag{6.8}$$

is low compared to computing the search direction itself. In special cases it may be even possible to solve this 1-D problem analytically.

Most line searches used in practice are inexact. The step length is chosen to approximately minimise f along the ray $\{x + \gamma d \mid \gamma \geq 0\}$, or even to just reduce f 'enough'.

A popular inexact algorithm that is simple and effective is called backtracking line search. It depends on two constants $0 < \alpha < \frac{1}{2}$, $0 < \beta < 1$.

Algorithm 6.2 (Backtracking line search)

- *give a descent direction d for f at x, $\alpha \in (0, \frac{1}{2})$, $\beta \in (0, 1)$*
- *set* $\gamma := 1$
- *while* $f(x + \gamma d) > f(x) + \alpha\gamma \nabla f(x) \cdot d$ *do* $\gamma := \beta\gamma$.

The algorithm employs a convexity assumption. From convexity we know that

$$\nabla f(x) \cdot (y - x) \geq 0 \quad \Rightarrow \quad f(y) \geq f(x), \tag{6.9}$$

so the search direction in a descent method must satisfy

$$\nabla f(x) \cdot d \quad < \quad 0, \tag{6.10}$$

where '$\cdot$' denotes the Euclidean scalar product. This means d must make an acute angle ('spitzer Winkel') with $-\nabla f(x)$. We call such a direction d with (6.10) a descent direction.

The line search algorithms above is called backtracking because it starts with unit step size and then reduces it by a factor β until the stopping condition

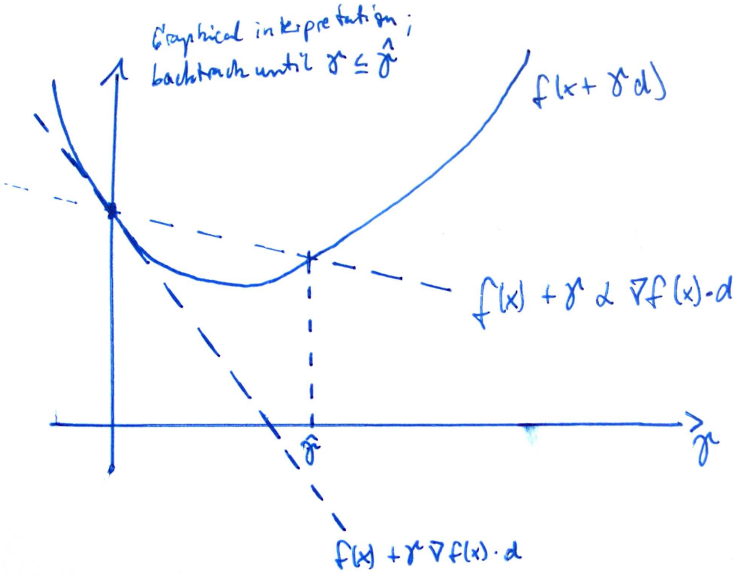
$$f(x + \gamma \Delta x) \quad \leq \quad f(x) + \alpha\gamma \nabla f(x) \cdot d \tag{6.11}$$

holds. Since d is a descent direction, (6.10) applies, so for small angle γ we have

$$\begin{array}{rcll} f(x + \gamma d) & \approx & f(x) + \gamma \nabla f(x) \cdot d & \text{(linearisation !)} \\ & \underset{\alpha \in (0, \frac{1}{2})}{<} & f(x) + \alpha\gamma \nabla f(x) \cdot d & \end{array} \tag{6.12}$$

which shows that the algorithm eventually terminates. The constant α can be interpreted as the fraction of the decrease in f predicted by linear extrapolation that we will accept, see sketch.

Sketch



Typical parameter choices in the literature are

$$0.01 \leq \alpha \leq 0.3 \quad \text{and} \quad 0.1 \leq \beta \leq 0.8. \tag{6.13}$$

Remark: The derivation of the scheme also works just by relying on $-\nabla f(x)$ as the steepest descent direction. By making use of (6.9) we see that there is a relation to the convexity of f.

7 Basic Order Line Search

Recall the following minimisation task: We seek a minimising sequence $x^{(k)}$ where

$$x^{(k+1)} \quad = \quad x^{(k)} + \gamma_k d^{(k)},$$

and where $\gamma_k > 0$, except when $x^{(k)}$ is optimal. In (6.6), we denote

- $d^{(k)} \in \mathbb{R}^n$ as the search direction
- $\gamma_k \in \mathbb{R}$ as the step size, or step length.

In a descent method we have

$$f(x^{(k+1)}) \quad < \quad f(x^{(k)}),$$

except when $x^{(k)}$ is optimal.

Motivation

By line search, straightforward algorithms can be constructed for many problems in Visual Computing. Also, it is a building block in many more advanced methods.

In this paragraph, we aim to fix useful search directions d of the general descent algorithm, complementing §6.

Mathematical Tool: We will rely very much on Taylor series expansions for functions of more than one variable. This works as follows. Given a function $f(x, y)$, with $f : \mathbb{R}^2 \to \mathbb{R}$, the Taylor series to second order about the point $(a, b)^\top$ is:

$$\begin{aligned} f(x,y) \quad &\approx \quad f(a,b) + (x-a)f_x(a,b) + (y-b)f_y(a,b) \\ &+ \quad \frac{1}{2}[(x-a)^2 f_{xx}(a,b) + 2(x-a)(y-b)f_{xy}(a,b) + (y-b)^2 f_{yy}(a,b)] \end{aligned} \tag{7.1}$$

We neglected higher order terms in (7.1). Using $\vec{x} := (x, y)^\top$ and $\vec{a} := (a, b)^\top$, we can write (7.1) in a more compact format:

$$f(\vec{x}) \quad \approx \quad f(\vec{a}) + (\vec{x} - \vec{a})^\top Df(\vec{a}) + \frac{1}{2}(\vec{x} - \vec{a})^\top D^2 f(\vec{a})(\vec{x} - \vec{a}). \tag{7.2}$$

Thereby, $Df(\vec{a})$ and $D^2f(\vec{a})$ denote the gradient and the (symmetric) Hessian matrix of f, respectively:

$$Df(\vec{a}) \equiv \nabla f(\vec{a}) = \begin{pmatrix} \frac{\partial}{\partial x} f(\vec{a}) \\ \frac{\partial}{\partial y} f(\vec{a}) \end{pmatrix}, \tag{7.3}$$

$$D^2f(\vec{a}) = Hf(\vec{a}) := \begin{pmatrix} \frac{\partial^2}{\partial x^2} f(\vec{a}) & \frac{\partial^2}{\partial x \partial y} f(\vec{a}) \\ \frac{\partial^2}{\partial y \partial x} f(\vec{a}) & \frac{\partial^2}{\partial y^2} f(\vec{a}) \end{pmatrix}$$

This procedure can be extended to $f : \mathbb{R}^n \to \mathbb{R}$ straightforwardly.

We will often use the compact form to write down formulae. Also, the use of the Hessian matrix allows useful theoretical investigations.

The Gradient Descent Method

If $f : \mathbb{R}^n \to \mathbb{R}$, $\vec{x} = (x_1, \ldots, x_n) \mapsto f(\vec{x})$, is defined and differentiable in a neighbourhood of $\vec{a} \in \mathbb{R}^n$, then f decreases fastest if one goes from $\vec{a}$ in the direction of the negative gradient of f in $\vec{a}$: $-\nabla f(\vec{a})$.

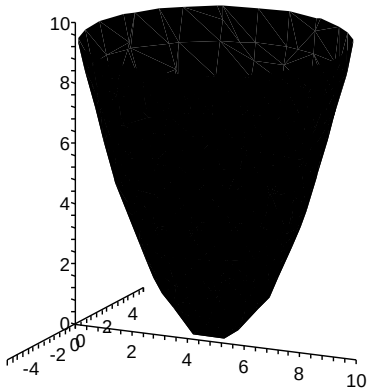
Thus, if we set

$$\vec{b} := \vec{a} - \gamma \nabla f(\vec{a}) \tag{7.4}$$

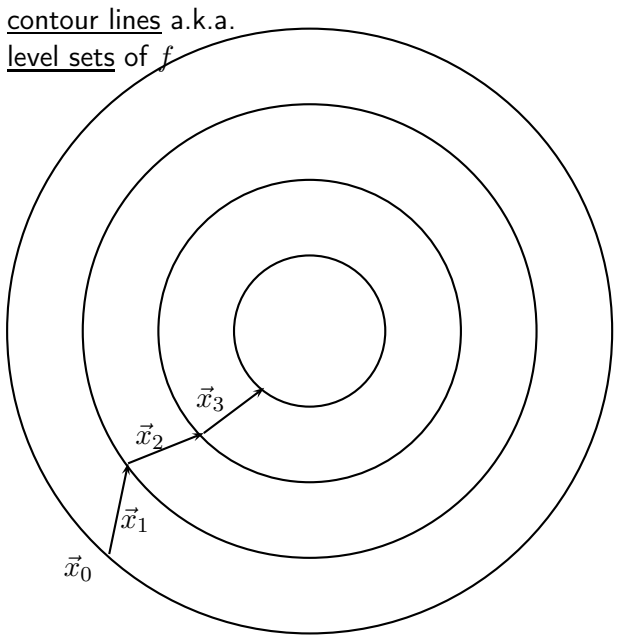
for $\gamma > 0$ a small enough number, then $f(\vec{b}) \leq f(\vec{a})$. Implementing this idea as an iterative scheme, the gradient descent method defines a sequence $\vec{x}_0, \vec{x}_1, \ldots$, with

$$\vec{x}_{k+1} = \vec{x}_k - \gamma_k \nabla f(\vec{x}_k), \qquad k = 0, 1, 2, \ldots. \tag{7.5}$$

We then have $f(\vec{x}_0) \geq f(\vec{x}_1) \geq f(\vec{x}_2) \geq \ldots$, so hopefully $(\vec{x}_k)_{k \geq 0}$ converges to the desired (local) minimum. Note, that γ_k is allowed to change every iteration.



Sketch:



Remarks

(i) Obviously, if f is strictly convex, (7.5) yields the global minimum.

(ii) At a local minimum, $\nabla f = \vec{0}$, the iteration (7.5) becomes stationary.

Relation to variational problems, Part I

It is easiest to see that we actually have employed a gradient descent method by interpreting the result of §5.1. Consider $E_\Delta(u)$ as a scalar-valued function of $u =$

$(u_1, \ldots, u_N)^\top$. Then the above methodology reads as

$$u^{k+1} \quad = \quad u_k - \gamma_k \nabla_u E_\Delta(u), \tag{7.6}$$

where

$$\nabla_u E_\Delta(u) \quad = \quad \left(\frac{\partial}{\partial u_1} E_\Delta(u), \ldots, \frac{\partial}{\partial u_N} E_\Delta(u) \right)^\top \tag{7.7}$$

Comparing (7.6)-(7.7) with (5.5)-(5.6a) we see that the latter can be written in the form of a gradient descent method.

Algorithm 7.1 (Gradient descent line search)

- *give a starting point* x
- *repeat*
  - *(a)* $d := -\nabla f(x)$
  - *(b) Ray search. Choose step size* γ*.*
  - *(c) Update.* $x := x + \gamma d$

  until stopping criterion is satisfied.

7.1 Newton's method

The well-known Newton method for finding zeros of a 'completely scalar function' can also be employed for minimising a function $f : \mathbb{R}^n \to \mathbb{R}$. This works as described in this paragraph.

Employing the Taylor expansion of $f(x)$ in a point $x^{(k)} \in \mathbb{R}^n$ leads to the approximate equality

$$\underbrace{f(x)}_{\in\mathbb{R}} = \underbrace{f(x^{(k)})}_{\in\mathbb{R}} + \underbrace{\nabla f(x^{(k)})}_{\in\mathbb{R}^n} \cdot \underbrace{(x - x^{(k)})}_{\in\mathbb{R}^n} + \frac{1}{2} \underbrace{(x - x^{(k)})}_{\in\mathbb{R}^n} \cdot \underbrace{[\underbrace{Hf(x^{(k)})}_{\in\mathbb{R}^{n\times n}} \underbrace{(x - x^{(k)})}_{\in\mathbb{R}^n}]}_{\in\mathbb{R}^n} \tag{7.8}$$

Let us stress, that $x^{(k)} \in \mathbb{R}^n$ is a **fixed**, chosen point, so that $f(x^{(k)})$, $\nabla f(x^{(k)})$ and $Hf(x^{(k)})$ are constant in (7.8).

Differentiating (7.8), i.e. applying the ∇-operator on both sides of the equation, we obtain

$$\nabla f(x) \quad = \quad \nabla f(x^{(k)}) + Hf(x^{(k)}) \cdot (x - x^{(k)}). \tag{7.9}$$

Enforcing the necessary optimality condition $\nabla f(x^*) = 0$ in a local minimum $x = x^*$, (7.9) gives

$$Hf(x^{(k)})(x - x^{(k)}) \quad = \quad -\nabla f(x^{(k)}), \tag{7.10}$$

i.e.,

$$x \;=\; x^{(k)} - [Hf(x^{(k)})]^{-1}\nabla f(x^{(k)}). \tag{7.11}$$

As we neglected higher order terms in (7.8)-(7.11), the point $x \in \mathbb{R}^n$ computed via (7.11) only approximates the minimum, but we can employ it in an iterative procedure:

$$x^{(k+1)} \;=\; x^{(k)} - [Hf(x^{(k)})]^{-1}\nabla f(x^{(k)}). \tag{7.12}$$

The method (7.12) is sometimes called Newton's scheme without stepsize control.

Comparing (7.12) to (6.6), we can interprete Newton's method as a line searching algorithm with

$$d^{(k)} \;:=\; -[Hf(x^{(k)})]^{-1}\nabla f(x^{(k)}). \tag{7.13}$$

Because of Hf, it is a second order line searching method.

Algorithm 7.2 (Newton's method)

- *give a starting point* x
- *repeat*
  - *(a)* $d := -[Hf(x)]^{-1}\nabla f(x)$.
  - *(b) Ray search. Choose step size* γ.
  - *(c) Update.* $x := x + \gamma d$

  until stopping criterion is satisfied.

Remark: Although we cast Newton's method by (7.12)-(7.13) in the format of a descent scheme, the related property (6.10) is not automatically satisfied.

8 The Trust Region Method

Motivation

Is there a line search scheme for 'difficult' optimisation problems?

The strictly convex problem solved by Newton's method is given by minimising

$$q(x) := f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (x - x^{(k)}) + \frac{1}{2}(x - x^{(k)}) \cdot Hf(x^{(k)})(x - x^{(k)}) \quad (8.1)$$

Strict convexity means: $Hf(x^{(k)})$ needs to be positive definite (PD).

Question: What if the Hessian Hf is not PD ?

We recall the condition for having a descent direction by a vector $\vec{s}$:

$$\nabla f(x) \cdot \vec{s}(x) \quad < \quad 0. \quad (8.2)$$

Now, let $\vec{s} := -[Hf]^{-1}\nabla f$ as in Newton's method. Then $\vec{s}$ is a descent direction, i.e.

$$\nabla f \cdot \vec{s} \quad = \quad -\nabla f \cdot [Hf]^{-1}\nabla f \overset{!}{<} 0, \quad (8.3)$$

if and only if $[Hf]^{-1}$ (and thus Hf) is PD. That is why a PD Hessian is needed for Newton's method.

Important observation by (8.3): Any PD matrix A with

$$\vec{s} \quad := \quad -A\nabla f \quad (8.4)$$

gives a descent direction.

We make use of this observation as follows. If the Hessian in $q(x)$ from (8.1) is not PD then we perturb it to be PD, and we minimise

$$Q(x) = f(x^{(k)}) + \nabla f(x^{(k)}) \cdot (x - x^{(k)}) + \frac{1}{2}(x - x^{(k)}) \cdot (Hf(x^{(k)}) + \alpha I)(x - x^{(k)}), \quad (8.5)$$

where I is the identity matrix. One easily derives

$$Q(x) \quad = \quad q(x) + \frac{\alpha}{2}\|x - x^{(k)}\|_2^2 \quad (8.6)$$

and accordingly the descent direction of the Trust Region (TR) method is

$$s^{(k)}(\alpha) \quad = \quad -[Hf(x^{(k)}) + \alpha I]^{-1} \nabla f(x^{(k)}) \tag{8.7}$$

Then $x^{(k)} + s^{(k)}(\alpha)$ minimises $Q(x)$, but we want a decrease in $q(x)$ as well, since $q(x)$ is nothing else but the Taylor expansion up to second order of our objective function f.

We have:

$$\begin{aligned}
& Q(x^{(k)} + s^{(k)}(\alpha)) && \leq Q(x) \\
& && \text{(known to be true!)} \\
\overset{(8.6)}{\Leftrightarrow} \quad & q(x^{(k)} + s^{(k)}(\alpha)) + \tfrac{\alpha}{2}\|s^{(k)}(\alpha)\|_2^2 && \leq q(x) + \frac{\alpha}{2}\|x - x^{(k)}\|_2^2, \\
& && \text{(true by eq.)} \qquad (8.8) \\
\Leftrightarrow \quad & q(x^{(k)} + s^{(k)}(\alpha)) + \tfrac{\alpha}{2}(\|s^{(k)}(\alpha)\|_2^2 - \|x - x^{(k)}\|_2^2) && \leq q(x) \qquad (8.9) \\
& && \text{(true by equivalence)}
\end{aligned}$$

The latter inequality implies that

$$q(x^{(k)} + s^{(k)}(\alpha)) \quad \leq \quad q(x) \tag{8.10}$$

in case of

$$\|s^{(k)}(\alpha)\|_2^2 - \frac{\alpha}{2}\|x - x^{(k)}\|_2^2 \quad \geq \quad 0, \tag{8.11}$$

i.e. if

$$\|x - x^{(k)}\|_2 \quad \leq \quad \|s^{(k)}(\alpha)\|_2 =: r_\alpha. \tag{8.12}$$

We have to understand (8.10)-(8.12) as follows. We aim to minimise $q(x)$, and we can do this by using $x^{(k)} + s^{(k)}(\alpha)$, see (8.10). However, as the condition (8.12) shows, this minimising effect only holds within a certain neighbourhood x of $x^{(k)}$. This is the "trust region".

Note that $x^{(k)}$ is just the current iterate in the k-th step, and we can employ such a construction for all k.

Let us discuss r_α from (8.12).

Lemma 8.1 *For suitable α, r_α is a non-increasing function of α.*

Proof:

Let $z_1, \ldots, z_n$ be an orthonormal eigenvector system of $Hf(x^{(k)})$ and where the ordered eigenvalues of $Hf(x^{(k)})$ are $\mu_1, \ldots, \mu_n$. Further, let

$$\nabla f(x^{(k)}) \quad = \quad \sum_{i=1}^{n} c_i z_i. \tag{8.13}$$

Let us remember, that the equation determining eigenvalues/-vectors shows:

$$Ax = \lambda x \quad \Leftrightarrow \qquad x = \lambda A^{-1}x \quad \Leftrightarrow \qquad A^{-1}x = \frac{1}{\lambda}x, \tag{8.14}$$

i.e. A and A^{-1} have the same eigenvectors with reciprocal eigenvalues. In the same way we obtain

$$(Hf + \alpha I)z_i = (\mu_i + \alpha)z_i \quad \Leftrightarrow \quad (Hf + \alpha I)^{-1}z_i = \frac{1}{\mu_i + \alpha}z_i \tag{8.15}$$

Thus:

$$\begin{aligned}
r_\alpha = \|s^{(k)}(\alpha)\|_2 \qquad &= \qquad \| - [Hf(x^{(k)}) + \alpha I]^{-1}\nabla f(x^{(k)})\|_2 \\
&\overset{(8.13)}{=} \qquad |-1| \left\| \sum_{i=1}^{n} c_i [Hf(x^{(k)}) + \alpha I]^{-1} z_i \right\|_2 \\
&\overset{(8.15)}{=} \qquad \left\| \sum_{i=1}^{n} \frac{c_i}{\mu_i + \alpha} z_i \right\|_2 \\
&= \qquad \sqrt{\left(\sum_{i=1}^{n} \frac{c_i}{\mu_i + \alpha} z_i \right) \cdot \left(\sum_{i=1}^{n} \frac{c_i}{\mu_i + \alpha} z_i \right)} \\
&\overset{\substack{(z_i \\ \text{orthonormal} \\ \text{basis})}}{=} \qquad \sqrt{\sum_{i=1}^{n} \frac{|c_i|^2}{|\mu_i + \alpha|^2}},
\end{aligned}$$

i.e. in summary

$$r_\alpha \quad = \quad \sqrt{\sum_{i=1}^{n} \frac{|c_i|^2}{|\mu_i + \alpha|^2}}. \tag{8.16}$$

For suitable α, which means $\mu_i + \alpha > 0$ for all $i = 1, \ldots, n$ must hold, (8.16) is clearly non-increasing in α

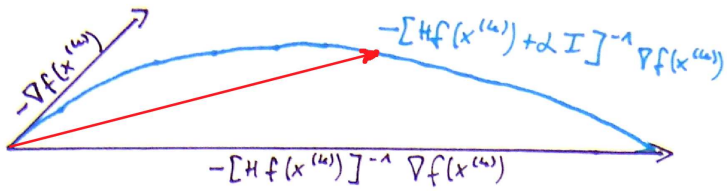
□.

Remarks:

(i) The bigger the α, the smaller the step.

(ii) Note that the condition $\mu_i + \alpha > 0$ implies that $Hf(x^{(k)}) + \alpha I$ is PD.

Sketch



The TR direction is a compromise between the gradient and Newton direction. If $\alpha = 0$ we have the Newton step, as $\alpha \to \infty$ we approach a small multiple of $-\nabla f(x^{(k)})$.

Summary: If the objective function $f(x)$ is not strictly convex or even non-convex, or if the Hessian, is ill-conditioned, the TR method should be used.

9 Convexity of Objective Functions

Motivation

Can we assess useful properties of optimisation schemes?

We will assume, that the objective function $f(x)$ is strictly convex. This assumption seems to be reasonable, since many algorithms rely on 'convexifying' the original problem.

We recall the optimality condition

$$\nabla f(x^*) \quad = \quad \vec{0}, \tag{9.1}$$

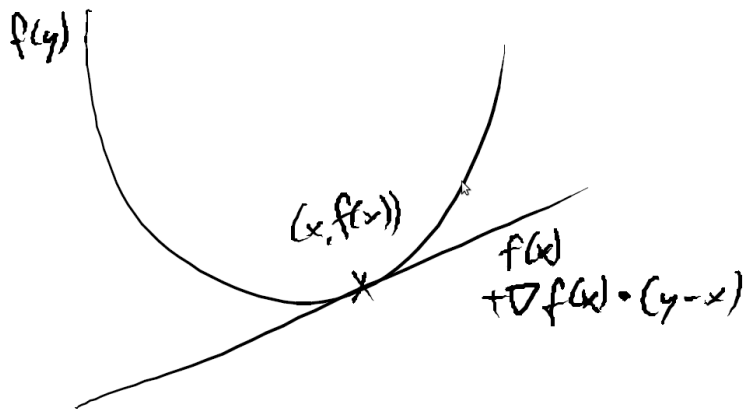
where $p^* = f(x^*)$ is optimal. We distinguish convexity conditions of first and second order for a point set $x \in D$.

9.1 First order convexity condition

The function f is convex on S if and only if

$$f(y) \geq f(x) + \nabla f(x) \bullet (y - x) \qquad \text{for all } x, y \in S, \tag{9.2}$$

see sketch.



The meaning of (9.2) is that we can infer from *local information* about a convex function (i.e. from its value $f(x)$ and derivative $\nabla f(x)$) a *global information* over S,

i.e.

$$f(x) + \nabla f(x) \bullet (y - x) \tag{9.3}$$

is a global underestimator for all $y \in S$. In particular, if $\nabla f(x) = 0$, (9.2) shows that $f(y) \geq f(x)$ for all $y \in S$.

9.2 Second order convexity condition

The geometrix requirement that f has a non-negative curvature if it is convex is encoded in the PSD (positive semidefinite) property of the Hessian matrix:

$$Hf(x) \succeq 0 \qquad \text{for all } x \in S. \tag{9.4}$$

In the case of strict convexity and Hf being PD (positive definite), it even holds

$$Hf(x) \succeq c_1 I \qquad \text{for all } x \in S, \tag{9.5}$$

where $c_1 > 0$.

9.3 Consequences of Convexity Conditions

Let us recall the Taylor series expansion

$$f(y) = f(x) + \nabla f(x) \bullet (y - x) + \frac{1}{2}(y - x) \bullet Hf(x)(y - x) + \dots \tag{9.6}$$

which one can make specific as by the formula for the remainder with $z \in \{x+t(y-x) \mid t \in (0,1)\}$

$$f(y) = f(x) + \nabla f(x) \bullet (y - x) + \frac{1}{2}(y - x) \bullet Hf(x)(y - x) \tag{9.7}$$

for some z on the line segment $[x, y]$. By (9.5), the last term is at least $\frac{c_1}{2}||y - x||_2^2$, so that

$$f(y) \geq f(x) + \nabla f(x) \bullet (y - x) + \frac{c_1}{2}||y - x||_2^2. \tag{9.8}$$

For $c_1 = 0$, we recover the first order condition (9.2). For $c_1 > 0$, we obtain a better lower bound on $f(y)$ than by (9.3).

We now show that the inequality (9.8) can be used to bound $f(x) - p^*$, and this is the suboptimality of the point x in terms of $||\nabla f(x)||_2$. We consider the right hand side of (9.8). Setting the gradient with respect to y equal to zero, we find that

$$\tilde{y} = x - \frac{1}{c_1}\nabla f(x) \tag{9.9}$$

minimises it. Therefore we have

$$\begin{aligned} f(y) &\geq f(x) + \nabla f(x) \bullet (y-x) + \frac{c_1}{2}\|y-x\|_2^2 \\ &\geq f(x) + \nabla f(x) \bullet (\tilde{y}-x) + \frac{c_1}{2}\|\tilde{y}-x\|_2^2 \\ &\overset{(9.9)}{=} f(x) - \underbrace{\nabla f(x) \bullet \left(\frac{1}{c_1}\nabla f(x)\right)}_{=-\frac{1}{c_1}\|\nabla f(x)\|_2^2} + \frac{c_1}{2}|-1| \cdot \underbrace{\left\|\frac{1}{c_1}\nabla f(x)\right\|_2^2}_{\frac{1}{c_1^2}\|\nabla f(x)\|_2^2}, \end{aligned}$$

i.e.

$$f(y) \geq f(x) - \frac{1}{2c_1}\|\nabla f(x)\|_2^2. \tag{9.10}$$

Since this holds for any $y \in S$, we have

$$p^* \geq f(x) - \frac{1}{2c_1}\|\nabla f(x)\|_2^2. \tag{9.11}$$

The latter inequality shows that if $\nabla f(x)$ is small at a point, then the point is nearly optimal. Thus, (9.11) can be interpreted as a condition for suboptimality which generalises (9.1):

$$\|\nabla f(x)\|_2^2 \leq (2c_1 \cdot \varepsilon)^{\frac{1}{2}} \Rightarrow f(x) - p^* \leq \varepsilon. \tag{9.12}$$

Remark

One may use (9.12) as a stopping criterion. However, c_1 is in general not known. We can only infer from (9.12) that for $||\nabla f(x)||_2 \leq \eta$, where η is (very likely) smaller than $(2c_1 \cdot \varepsilon)^{\frac{1}{2}}$, then we have $f(x) - p^* \leq \varepsilon$ (very likely).

10 Convex Functions and Convex Sets

10.1 Condition number of sublevel sets

Recall the convexity conditions

$$f(y) \geq f(x) + \nabla f(x) \bullet (y - x) \qquad \text{for all } x, y \in S, \tag{10.1}$$

and

$$Hf(x) \succeq c_1 I \qquad \text{for all } x \in S \tag{10.2}$$

plus the resulting inequality

$$f(y) \geq f(x) + \nabla f(x) \bullet (y - x) + \frac{c_1}{2}||y - x||_2^2. \tag{10.3}$$

Strong Convexity Effects

Assuming that methods of interest are descent methods making use of a starting point $x^{(0)}$, we introduce the sublevel set

$$S \;=\; \{x \in D \mid f(x) \leq f(x^{(0)})\}, \tag{10.4}$$

on which our methods act.

The inequality of (10.3) implies that the sublevel sets contained in S are bounded, i.e. also S is bounded. Therefore the maximum eigenvalue of $Hf(x)$, which is a continuous function of x on S, is bounded via

$$Hf(x) \preceq c_2 I \qquad \text{for all } x \in S, \tag{10.5}$$

where $c_2 > c_1$ is a constant. Summarising (10.2) and (10.5), we have

$$c_1 I \preceq Hf(x) \preceq c_2 I \qquad \text{for all } x \in S. \tag{10.6}$$

The ratio $\kappa := \frac{c_2}{c_1}$ is thus an upper bound on the condition number of the matrix $Hf(x)$, i.e. on the ratio of its largest eigenvalue to its smallest eigenvalue.

We can also interpret (10.6) geometrically in terms of the sublevel sets of f, which we show now.

We define the width of a convex set $C \subseteq \mathbb{R}^n$, in the direction q, where $\|q\|_2 = 1$, as

$$W(C, q) := \sup_{z \in C} q \bullet z - \inf_{z \in C} q \bullet z. \tag{10.7}$$

The minimum and maximum width of C are given by

$$W_{\min} := \inf_{\|q\|_2 = 1} W(C, q), \qquad W_{\max} := \sup_{\|q\|_2 = 1} W(C, q). \tag{10.8}$$

The condition number of the convex set C is then defined as

$$\operatorname{cond}(C) = \frac{W_{\max}^2}{W_{\min}^2}, \tag{10.9}$$

i.e. the square of the ratio is its maximum width to its minimum width.

The number $\operatorname{cond}(C)$ measures its anisotropy or eccentricity. If it is near one, then C is nearly spherical, if it is large, then C is far wider in one direction than in others.

We will now derive a bound on the condition number of the α-sublevel set

$$C_\alpha = \{x | f(x) \leq \alpha\}, \qquad \text{where } p^* < \alpha < f(x^{(0)}). \tag{10.10}$$

By (10.5), we get in the same way that lead to (10.3) the estimate

$$f(y) \leq f(x) + \nabla f(x) \bullet (y - x) + \frac{c_2}{2}\|y - x\|_2^2. \tag{10.11}$$

Together with (10.3), for $x = x^*$ we have

$$p^* + \frac{c_2}{2}\|y - x^*\|_2^2 \overset{(10.11)}{\geq} f(y) \overset{(10.11)}{\geq} p^* \frac{c_1}{2}\|y - x^*\|_2^2. \tag{10.12}$$

This shows, that the α-sublevel set contains a ball B_{inner}, and is contained in a ball B_{outer}, with radii

$$r_{\text{inner}} = \sqrt{\frac{2}{c_2}(\alpha - p^*)}, \qquad r_{\text{outer}} = \sqrt{\frac{2}{c_1}(\alpha - p^*)}, \tag{10.13}$$

respectively. The ratio of the radii squared gives an upper bound on the condition number of C_α:

$$\operatorname{cond}(C_\alpha) \leq \frac{c_2}{c_1}. \tag{10.14}$$

We can now give a geometric interpretation of the condition number of the Hessian matrix at the optimum. From the Taylor series expansion of f around x^*,

$$f(y) \approx p^* + \frac{1}{2}(y - x^*) \bullet Hf(x^*)(y - x^*), \tag{10.15}$$

we obtain for α close to p^*

$$C_\alpha \approx \{y \mid (y - x^*) \bullet Hf(x^*)(y - x^*) \leqslant 2(\alpha - p^*)\}, \tag{10.16}$$

i.e. the sublevel set is well approximated by an ellipsoid with center at x^*. Therefore

$$\lim_{\alpha \to p^*} \operatorname{cond}(C_\alpha) = \operatorname{cond}(Hf(x^*)). \tag{10.17}$$

We will see that the condition number of the sublevel sets of f has a strong effect on the efficiency of schemes.

10.2 Analysis of Gradient Descent

We consider for simplicity the gradient descent scheme with an exact line search. The scheme then reads as

$$x^{(k+1)} = x^{(k)} + \gamma^{(k)}(-\nabla f(x^{(k)})). \tag{10.18}$$

Recalling (10.11), i.e.

$$f(y) \leq f(x) + \nabla f(x) \bullet (y - x) + \frac{c_2}{2}\|y - x\|_2^2, \tag{10.19}$$

we obtain for $x := x^{(k)}$ and $y := x^{(k)} - \gamma \nabla f(x^{(k)})$ a quadratic upper bound on $\tilde{f}(\gamma) := f(x^{(k)} - \gamma \nabla f(x^{(k)}))$:

$$\tilde{f}(\gamma) \leq f(x^{(k)}) - \gamma \|\nabla f(x^{(k)})\|_2^2 + \frac{c_2 \cdot \gamma^2}{2}\|\nabla f(x^{(k)})\|_2^2. \tag{10.20}$$

Minimising both sides with respect to γ gives

$$f(x^{(k+1)}) = \tilde{f}(\gamma_{\text{exact}}) \leq f(x^{(k)}) - \frac{1}{2c_2}\|\nabla f(x^{(k)})\|_2^2. \tag{10.21}$$

Subtracting p^* from both sides gives

$$f(x^{(k+1)}) - p^* \leq f(x^{(k)}) - p^* - \frac{1}{2c_2}\|\nabla f(x^{(k)})\|_2^2. \tag{10.22}$$

Employing then the suboptimality condition (9.8) in the form

$$\|\nabla f(x^{(k)})\|_2^2 \geq 2c_1(f(x^{(k)}) - p^*) \tag{10.23}$$

we obtain from (10.22):

$$f(x^{(k+1)}) - p^* \leq (1 - \frac{c_1}{c_2})(f(x^{(k)}) - p^*). \tag{10.24}$$

With

$$c := 1 - \frac{c_1}{c_2} < 1 \tag{10.25}$$

we get by a recursive application of (10.24):

$$f(x^{(k)}) - p^* \leq \underbrace{c \cdot \ldots \cdot c}_{k\text{times}} \cdot (f(x^{(0)}) - p^*) \tag{10.26}$$

which shows the convergence of the method. In particular, we must have $f(x^{(k)}) - p^* \leq \varepsilon$ if

$$c^k (f(x^{(0)}) - p^*) \stackrel{!}{\leq} \varepsilon. \tag{10.27}$$

Solving (10.27) for k shows that at most

$$\frac{\log(\frac{f(x^{(0)}) - p^*}{\varepsilon})}{\log(\frac{1}{c})} \tag{10.28}$$

iterations are needed.

Let us remark that the numerator in (10.28)

$$\log \frac{f(x^{(0)}) - p^*}{\varepsilon} \tag{10.29}$$

can be interpreted as the $\log$ of the ratio of the initial suboptimality (i.e. ε). This term suggests that the number of iterations depends on the starting point.

The denominator $\log(\frac{1}{c})$ is a function of $\frac{c_2}{c_1}$, see (10.25). Thus it is related to the bound on the condition number of the matrix $Hf(x)$ over S, or of the sublevel sets $\{z \mid f(z) \leq \alpha\}$, respectively. For a large condition number bound $\frac{c_2}{c_1}$, we have

$$\log(\frac{1}{c}) \quad = \quad -\log\left(1 - \frac{c_1}{c_2}\right) \approx \frac{c_1}{c_2}, \tag{10.30}$$

so that the number of iterations increases approximately linearly with increasing $\frac{c_2}{c_1}$.

Summary

- The efficiency of a stopping criterion depends on the curvature of the objective function.
- The efficiency of a scheme depends on the relation of the search direction to the shape of sublevel sets of the objective function.

11 Constrained Optimisation

Motivation

How can we deal with constraints in a minimizing process?

Mathematical model of the underlying problem ("nonlinear programming problem (NLPP)")

$$\begin{cases} \text{Minimise} f(x) \\ \text{subject to the constraints} \\ g_i(x) \leq 0, i = 1, \ \ldots, m; \\ h_j(x) = 0, \ j = 1, \ldots, p \end{cases} \tag{11.1}$$

Given constraints have to be transformed in advance into the formats $g_i(x) \leq 0$ for inequality constraints, and $h_j(x) = 0$ for equality constraints, respectively.

Two main strategies are available:

- *use* information from optimality conditions
  $\Rightarrow$ techniques: dual method, augmented Lagrangian algorithms
- *not use* information from optimality conditions
  $\Rightarrow$ techniques: penalisation and barrier methods

Key idea:
Can we incorporate constraints into an easy-to-solve, unconstrained model, which delivers the solution of the constrained problem?

11.1 Penalisation and Barrier Methods

Idea: Infeasible values/vectors are prohibited, or at least penalised.

Basic set-up:

$$\begin{cases} \text{Minimise} & f(x) + \tilde{f}(x), \\ \text{where} & \tilde{f}(x) = \begin{cases} 0, & g_i(x) \leq 0,\ h_j(x) = 0,\ \forall i, j \\ +\infty, & \text{otherwise.} \end{cases} \end{cases} \tag{11.2}$$

Prohibited vectors $\hat{x}$ are eliminated in this set-up from the minimisation process via the contribution $\tilde{f}(\hat{x}) = +\infty$ in the cost function.

Problem statement:

$f + \tilde{f}$ is not continuous, as it can take on the value $+\infty$ abruptly.

Solution No.1 - Penalisation methods

We assign not an infinite cost to an infeasible vector, but we penalise it, thus „discouraging" it to be part of the solution.

One of the most popular families of penalisations is

$$\tilde{f}_r(x) \quad = \quad r \left(\sum_{i=1}^{m} \max\{0, g_i(x)\}^l + \sum_{j=1}^{l} |h_j(x)|^q \right), \tag{11.3}$$

where $l, q \geq 1$ are exponents and $r \geq 1$ is a penalisation parameter.

Remarks

- If $l > 1$, the function $\max\{0, g_i(x)\}^l$ is differentiable if $g_i(x)$ is.
- A typical setting is the quadratic penalisation corresponding to $l = q = 2$.

Solution No.2 - Barrier methods

Example for choosing $\tilde{f}$:

$$\tilde{f}_r(x) \quad = \quad -\frac{1}{r} \sum_{i=1}^{m} \frac{1}{g_i(x)}. \tag{11.4}$$

Moving toward the boundary of a feasible set, $g_i(x)$ becomes close to zero, and so $\tilde{f}_r(x)$ becomes large, placing a barrier at the boundary between feasible and infeasible vectors (where $\tilde{f}_r(x)$ formally reaches $+\infty$).

For constraints of the form $h_j(x) = 0$, one possibility is to add

$$r^3 \sum_{j=1}^{p} \frac{h_j(x)^2}{1 - r^2 h_j(x)^2} \tag{11.5}$$

to $\tilde{f}_r(x)$.

Penalisation vs. Barrier methods

- in penalisation methods, iterates of x are penalised outside the set of feasible vectors.
- in barrier methods, iterates of x are penalised inside the set of feasible points.
- barrier methods rely on a starting vector $x^{(0)}$ inside the feasible set.

11.2 Lagrange Multiplier

Let us consider the reduced problem

$$\text{Minimize } f(x) \text{ under } h(x) \;=\; 0. \tag{11.6}$$

Let us have a look at parametrised curves

$$\tau : (-\delta, \delta) \to \mathbb{R}^n, \qquad \delta > 0, \tag{11.7}$$

whose image $\tau(-\delta, \delta)$ is entirely contained in the feasible set of our optimisation problem, i.e.

$$h(\tau(t)) \;=\; 0, \qquad \forall t \in (-\delta, \delta) \tag{11.8}$$

If we suppose that $x_0 \in \mathbb{R}^n$ is a point of local minimum or maximum, or even a saddle point, and assume that τ passes through x_0 for $t = 0$, $\tau(0) = x_0$, **then** the composition $f(\tau(t))$ must likewise have a local extremum or saddle point for $t = 0$. In any of these cases, we must have a vanishing first derivative:

$$0 = \left.\frac{\mathrm{d} f(\tau(t))}{\mathrm{d}t}\right|_{t=0} \;=\; \underbrace{\nabla f(\tau(0))}_{\in \mathbb{R}^n} \cdot \underbrace{\tau'(0)}_{\mathbb{R}^n} = \nabla f(x_0) \cdot \tau'(0) \tag{11.9}$$

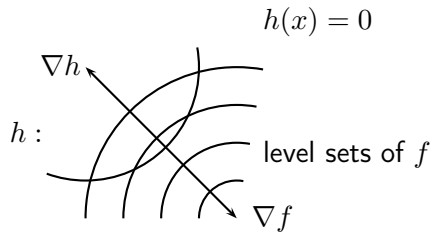
On the other hand, since $h(\tau(t)) \equiv 0$ for all t of interest, we should also have

$$0 \;=\; \nabla h(x_0) \cdot \tau'(0) \tag{11.10}$$

as the gradient of the zero function, see (11.8), is zero. Since the tangient vector $\tau'(0)$ is arbitrary, the equalities (11.9) and (11.10) can both hold if and only if $\nabla f(x_0)$ belongs to the span of $\nabla h(x_0)$, i.e.

$$\nabla f(x) + \lambda \nabla h(x) \;=\; 0, \tag{11.11}$$

and the pair (x, λ) consists of the unknowns.



Sketch

11.3 Augmented Lagrangian algorithms

We consider for motivation only the problem

$$\text{Minimise } f(x) \text{ under } h(x) = 0. \tag{11.12}$$

We know that solutions must satisfy

$$\nabla f(x) + \lambda \nabla h(x) \quad = \quad 0, \tag{11.13}$$

for an appropriate multiplier λ. Assuming that we have an approximate value λ_j, we now show how we can use this

- to find the corresponding optimal solution x_j, and to
- simultaneously improve the approximation of the multiplier to proceed iteratively to λ_{j+1}.

Starting with

$$\text{Minimise } f(x) + \lambda_j h(x) \text{ under } h(x) = 0, \tag{11.14}$$

(j iteration index) we introduce a quadratic penalizer and treat instead the problem

$$\text{Minimise } f(x) + \lambda_j h(x) + \frac{r_j}{2}|h(x)|^2 \tag{11.15}$$

for some parameter r_j. The optimality condition for the latter is

$$\nabla f(x) + \lambda_j \nabla h(x) + r_j h(x) \nabla h(x) \quad = \quad 0. \tag{11.16}$$

Assuming that x_j is „good“ , it must be close to the true optimal solution x.

Comparison of the optimality conditions (11.13) and (11.16) leads to

$$\lambda_j + r_j h(x_j) \quad \approx \quad \lambda. \tag{11.17}$$

With a number $0 < c < 1$, this idea yields the sought algorithm class using

$$\lambda_{j+1} \quad := \quad \lambda_j + r_j h(x_j), \qquad r_{j+1} := c r_j. \tag{11.18}$$

12 Duality in Constrained Optimization

Let us recall the model problem

$$\begin{cases} \text{Minimise} f(x) \\ \text{subject to the constraints} \\ g_i(x) \leq 0, i = 1, \ldots, m; \\ h_j(x) = 0, \ j = 1, \ldots, p \end{cases} \tag{12.1}$$

We define the Lagrangian

$$L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \ \rightarrow \ \mathbb{R} \tag{12.2}$$

associated with (12.1) as

$$L(\vec{x}, \vec{\mu}, \vec{\lambda}) \ = \ f(\vec{x}) + \sum_{i=1}^{m} \mu_i g_i(\vec{x}) + \sum_{j=1}^{p} \lambda_j \cdot h_j(\vec{x}), \tag{12.3}$$

where $\mu_i \geq 0$.

We refer to the μ_i and the λ_j as the Lagrangian multiplier or dual variables. We define the Lagrange dual function as

$$\Theta(\vec{\mu}, \vec{\lambda}) \ = \ \inf_{\vec{x}} L(\vec{x}, \vec{\mu}, \vec{\lambda}) \tag{12.4}$$

Remark: Since $\Theta(\vec{\mu}, \vec{\lambda})$ is the pointwise infimum of a family of *affine functions* of $(\vec{\mu}, \vec{\lambda})$, it can be shown to be convex, even if the original problem (12.1) is not convex.

The lower bound property

The dual function gives lower bounds on the optimal value $\vec{p}^*$ of (12.1). Suppose $\vec{x}^*$ is feasible, i.e. it satisfies the constraints, then

$$L(\vec{x}^*, \vec{\mu}, \vec{\lambda}) \ = \ \underbrace{f(\vec{x}^*)}_{=\vec{p}^*} + \sum_{i=1}^{m} \underbrace{\underbrace{\mu_i}_{\geq 0} \underbrace{g_i(\vec{x}^*)}_{\leq 0}}_{\leq 0} + \sum_{j=1}^{p} \lambda_j \underbrace{h_j(\vec{x}^*)}_{=0} \tag{12.5}$$

and thus

$$\Theta(\vec{\mu}, \vec{\lambda}) \quad \leq \quad p^* \tag{12.6}$$

This motivates

Definition 12.1 *The difference*

$$\min\{f(\vec{x}) \mid g_i(\vec{x}) \leq 0,\ h_j(\vec{x}) = 0\ \forall i, j\} - \max\{\Theta(\vec{\mu}, \vec{\lambda}) \mid \mu_i \geq 0\ \forall i, j\} \tag{12.7}$$

is called duality gap. When there is no such gap, the primal problem (12.1) *is equivalent to the dual problem*

$$\begin{cases} Maximise & \Theta(\vec{\mu}, \vec{\lambda}) \\ subject\ to & \mu_i \geq 0,\ i = 1, \ldots, m. \end{cases} \tag{12.8}$$

This situation is refered to as strong duality.

Remarks

(i) If f and g are convex and h is affine, the primal problem and the dual problem are equivalent.

(ii) One may try to use the lower bound property to construct a stopping criterion.

Linear approximation interpretation

Let us rewrite the primal problem (12.1):

$$\text{Minimise } f(\vec{x}) + \sum_{i=1}^{m} I_m(g_i(\vec{x})) + \sum_{j=1}^{p} I_0(h_j(\vec{x})), \tag{12.9}$$

where $I_m : \mathbb{R} \to \mathbb{R}$ is the indicator function for the non-positive reals,

$$I_m(a) \quad = \quad \begin{cases} 0, & a \leq 0 \\ +\infty & a > 0, \end{cases} \tag{12.10}$$

and similarly, I_0 is the indicator function of $\{0\}$. In the formulation (12.9), the function $I_m(a)$ expresses our displeasure associated with a constraint function value $a = g_i(\vec{x})$: It is zero, if $g_i(\vec{x}) \leq 0$, and infinite if $g_i(\vec{x}) > 0$.

Analogously, $I_0(a)$ gives our displeasure for a violated equality constraint $a = h_j(\vec{x})$.

These are „hard“ displeasure functions, since our displeasure rises directly from zero to infinity.

Now we replace in (12.9) the function $I_m(a)$ with the linear function $\mu_i \cdot a$, where $\mu_i \geq 0$ and $I_0(a)$ with $\lambda_j \cdot a$. The objective function in (12.9) then becomes the Lagrangian $L(\vec{x}, \vec{\mu}, \vec{\lambda})$, and the dual function is the optimal value of the problem

$$\text{Minimise } L(\vec{x}, \vec{\mu}, \vec{\lambda}). \tag{12.11}$$

Thus, the meaning of the Lagrangian involves a linear or „soft" displeasure function instead of $I_m(a)$ and $I_0(a)$. The displeasure grows as the constraints become more „more violated".

Note that unlike the original formulation, in which any x with $g_i(\vec{x}) \leq 0$ is just acceptable, in the soft formulation we actually derive pleasure from $g_i < 0$ as this contributes to minimise $L(x, \vec{\mu}, \vec{\lambda})$. That is why we need to maximise $\Theta(\vec{\mu}, \vec{\lambda})$ in (12.8).

12.1 Karush-Kuhn-Tucker (KKT) optimality conditions

Consider now the „full" problem (12.9). Starting from strong duality, i.e. with $g_i(\vec{x}) = 0$, we obtain at a local minimum $\vec{x}_0$

$$\nabla f(\vec{x}_0) + \sum_{i=1}^{m} \mu_i \nabla g_i(\vec{x}_0) + \sum_{j=1}^{p} \lambda_j \nabla h_j(\vec{x}_0) \quad = \quad 0. \tag{12.12}$$

Furthermore, $\mu_i \geq 0$ as usual. The intuitive reason is, that f shall attain a minimum at $\vec{x}_0$ but $g(\vec{x})$ has a maximum as $g(\vec{x}) = 0$ is the maximum value g should attain. Hence, ∇f and ∇g at $\vec{x}_0$ must „point in different directions".
This yields

Theorem 12.1 *If $\vec{x}$ is a optimal solution and strong duality holds, then there exists a vector of multipliers (μ, λ) such that*

$$\begin{cases} \nabla f(\vec{x}) + \sum_{i=1}^{m} \mu_i \nabla g_i(\vec{x}) + \sum_{j=1}^{p} \lambda_j \nabla h_j(\vec{x}) & = 0 \\ \mu_i g_i(\vec{x}) & = 0 \\ \mu_i \geq 0, g_i(\vec{x}) \leq 0, h(\vec{x}) & = 0. \end{cases} \tag{12.13}$$

These necessary conditions are the KKT conditions.

Remark: For m conditions $g_i(\vec{x}) \leq 0$ and p conditions $h_j(\vec{x}) = 0$, we thus obtain in $\mathbb{R}^n$ a system of $n + m + p$ equations in the $n + m + p$ unknowns $(\vec{x}, \vec{\mu}, \vec{\lambda})$.

Summary

- Using the Lagrangian, we can often simplify, or reformulate, a constrained optimisation problem.
- The dual problem can be used to derive alternative algorithms, construct stopping criteria, or for theoretical purposes.

13 The Bregman Iteration

Motivation

- How can one construct efficient methods for some difficult problems?
- Examples where Bregman iteration is used: TV denoising, deconvolution, compressed sensing, machine learning, ...

The tool investigated here is based on the notion of the Bregman distance, or Bregman divergence. To construct this, consider an objective function

$$\begin{cases} f : D \to \mathbb{R}, D \text{ some domain} \\ f \text{ strictly convex} \end{cases} \tag{13.1}$$

Let $x, y \in \mathbb{R}^n$ be two points in D. Linearising f via a Taylor series expansion around y gives

$$L_{f,y}(\xi) \quad := \quad f(y) + (\xi - y) \bullet \nabla f(y), \qquad \xi \in \mathbb{R}^n. \tag{13.2}$$

Evaluating $L_{f,y}$ at $x \in \mathbb{R}^n$ and comparing this linearised form with $f(x)$ gives the Bregman distance:

$$B_f(x, y) \quad = \quad f(x) - f(y) - (x - y) \bullet \nabla f(y). \tag{13.3}$$

One should be careful with this notion, as this distance is not symmetric: In general

$$B_f(x, y) \quad \neq \quad B_f(y, x). \tag{13.4}$$

13.1 Bregman Distance and Duality

The Bregman distance may find application in dual formulations of optimisation problems. The key notion in this context is the notion of a convex conjugate function.

For a function f taking values on the extended real numbers,

$$f : D \to \mathbb{R} \cup \{+\infty\}, \tag{13.5}$$

the convex conjugate function $f^*(x^*)$ is defined by

$$f^*(x^*) \quad := \quad \sup_x \{x^* \bullet x - f(x) | x \in D\} \tag{13.6}$$

or, equivalently by

$$f^*(x^*) \quad := \quad \inf_x \{f(x) - x^* \bullet x | x \in D\} \tag{13.7}$$

Remarks

a) This operation is order-reversing, i.e. if $f \leq g$ (pointwise), then $f^* \geq g^*$ (pointwise).

b) f^* is sometimes called the Fenchel-Legendre-transform of f.

We also rely on the notion of the proper convex function. A function $f : D \to \mathbb{R}$ satisfies this notion if it is convex and takes values in the extended real numbers such that

$$\begin{cases} f(x) < \infty & \text{for at least one } x, \textit{and} \\ f(x) > -\infty & \text{for every } x. \end{cases} \tag{13.8}$$

A proper convex function has the property that one can find $b \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ with

$$f(x) \geq x \cdot b - \beta \qquad \forall x. \tag{13.9}$$

Analogously, we can define proper concave functions.

Then one can show the following results:

Theorem 13.1 (Fenchel's duality theorem) *For f proper convex and g proper concave, it holds*

$$\min_x (f(x) - g(x)) \quad = \quad max_p (g^*(p) - f^*(p)) \tag{13.10}$$

Theorem 13.2 (Fenchel-Young inequality) *For any proper convex f holds*

$$p \cdot x \quad \leq \quad f(x) + f^*(p). \tag{13.11}$$

Concerning the Bregman distance, one can show the dual symmetry relation:

$$B_{f^*}(x^*, y^*) = B_f(y, x), \tag{13.12}$$

where $x^* = \nabla f(x)$ is the dual point corresponding to x.

13.2 Application to TV Denoising

We consider here the Rudin-Osher-Fatemi (ROF) model for total variation (TV) denoising, which has received much attention in the last years.

The mathematical model reads as

$$u \;=\; \underset{u}{\operatorname{argmin}} \int_{\Omega} |\nabla u| + \frac{\lambda}{2}\|Ku - f\|_2^2 d\vec{x}, \tag{13.13}$$

where $\vec{x} = (x_1, x_2)^\top$, and where Ω is the domain of a given image f, $\lambda \in \mathbb{R}$ is a parameter, and $K \in \mathbb{R}^{n\times n}$ is matrix that is often set as $K := I$ for TV denoising.

The part

$$\int_{\Omega} |\nabla u| d\vec{x} \tag{13.14}$$

where

$$|\nabla u| = \sum_{i=1}^{2} |\partial_{x_i} u|, \tag{13.15}$$

constitutes the total variation norm. The non-differentiability of the latter leads to significant numerical problems.

A useful way to deal with the non-differentiability is to introduce a new variable to separate the calculation of the non-differentiable term and the fidelity term. In this way, the model (13.13) can be made equivalent to

$$\begin{cases} u = \operatorname{argmin}_{(u,q)} \int_\Omega |q| + \frac{\lambda}{2}\|Ku - f\|_2^2 d\vec{x} \\ \text{under the contraint } q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} \partial_{x_1} u \\ \partial_{x_2} u \end{pmatrix} = \nabla u. \end{cases} \tag{13.16}$$

In order to make an unconstrained problem out of the constrained formulation (13.16), we add a penalty term, cf. §12, to the objective function:

$$u \;=\; \underset{(u,q)}{\operatorname{argmin}} \int_{\Omega} |q| + \frac{\lambda}{2}\|Ku - f\|_2^2 + \frac{\mu}{2}\|q - \nabla u\|_2^2 d\vec{x} \tag{13.17}$$

However, we need a way of modifying the problem (13.17) to get the exact enforcement of the constraint. Grouping the first two energy terms together,

$$E(u,p) \;:=\; \int_{\Omega} |q| + \frac{\lambda}{2}\|Ku - f\|_2^2 d\vec{x}, \tag{13.18}$$

we write

$$u \;=\; \underset{(u,q)}{\operatorname{argmin}}\; E(u,q) + \frac{\mu}{2}\int_{\Omega} \|q - \nabla u\|_2^2 d\vec{x}. \tag{13.19}$$

We now treat (u, q) as one vector, writing for formal purposes $\vec{\eta} := (u, q)^\top$. Then the Bregman distance of the convex functional $E(u, q) = E(\vec{\eta})$ reads as:

$$B_E(\vec{\eta}, \vec{\xi}) \;=\; E(\vec{\eta}) - E(\vec{\xi}) - \nabla_{\vec{\eta}} E(\vec{\xi}) \bullet (\vec{\eta} - \vec{\xi}). \tag{13.20}$$

Rather than solve (13.19), one may recursively solve

$$(u^{k+1}, q^{k+1}) \quad = \quad \underset{(u,q)}{\operatorname{argmin}} \; B_E((u,q),(u^k,q^k)) + \frac{\mu}{2} \int_\Omega \|q^k - \nabla u\|_2^2 d\vec{x}. \quad (13.21)$$

Remark

The benefit in having (13.21) is a higher regularity compared to the original form (13.13). The numerical advantages gained in this way are not obvious. However, one can validate:

- The resulting Bregman iteration converges very quickly.
- The value μ can be chosen as a constant, e.g. it can be chosen to optimise the convergence speed of the iterative algorithm.
- The Bregman iteration behaves very stable.

Special care need to be taken interpreting $\nabla_{\vec{\eta}} E(u^k, q^k)$ within $B_E((u,q),(u^k,q^k))$, as E is in the considered case not differentiable. However, E is (due to the integral over Ω) Lipschitz continuous which suffices to define subdifferentials.

The subdifferential is a generalisation of the gradient. For a function $f : \mathbb{R}^n \to \mathbb{R}$, the subdifferential at $\hat{x}$ is given by

$$\partial f(\hat{x}) = \{g \in \mathbb{R}^n : f(x) - f(\hat{x}) \geq g \bullet (x - \hat{x}) \quad \forall x \in \mathbb{R}^n\}. \quad (13.22)$$

The elements $g \in \partial f(\hat{x})$ are called subdifferentials.

Example

The subdifferential of the function $f : \mathbb{R} \to \mathbb{R}, f(x) = |x|$, is given by

$$\partial f(\hat{x}) \quad = \quad \begin{cases} -1 & , \hat{x} < 0, \\ [-1,1] & , \hat{x} = 0, \\ 1 & , \hat{x} > 0. \end{cases} \quad (13.23)$$

To conclude, $\nabla_{\vec{\eta}} E(u^k, q^k)$ is an element in $\partial E(u^k, q^k)$.

Remarks

a) It can be proven rigorously that the described strategy works.

b) One can get an explicit formula for $\nabla_{\vec{\eta}} E$, so that the construction of an iterative scheme is simple.

Summary

The use of the Bregman distance offers possibilities to reformulate difficult optimisation problems into simpler ones, also leading to efficient schemes.

14 Splitting Schemes

Motivation

Is it possible to simplify optimisation algorithms?

We begin by studying a prototype of variational problems:

$$u^* = \min_{u \in \mathcal{H}} I_C(u) + \frac{1}{2}||f - u||^2, \tag{14.1}$$

where

- C is a convex set of sought functions,
- $I_C(\cdot)$ is the indicator function:

$$I_C : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}, \qquad I_C(u) = \begin{cases} 0, & \text{if } u \in C \\ +\infty, & \text{otherwise} \end{cases} \tag{14.2}$$

- $\mathcal{H}$ is a real Hilbert space with an inner product $\langle \cdot, \cdot \rangle$ and a norm $||.||$, i.e. it is a vector space of functions where one can measure angles and distances.

The task of (14.1) is to determine the projection of f onto C, i.e. to find in a predefined set of functions C the closest one to f.

As in penalisation methods, we generalise now the function I_C. It turns out to be useful to consider the class of functions $\Gamma_0(\mathcal{H})$.

Definition 14.1 *An extended real-valued function $E : \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ belongs to $\Gamma_0(\mathcal{H})$ if E is*

(i) convex,

(ii) not identically equal to $+\infty$, and

(iii) lower semicontinuous (LSC).

Lower semicontinuity is a property of extended real-valued functions that is weaker than continuity. It means that for every 'point' $u_0 \in \mathcal{H}$, $E(u)$ is either close to or larger than $E(u_0)$ if u is in the neighbourhood of u_0.

The variational problem obtained by replacing I_C with an arbitrary function $E \in \Gamma_0(\mathcal{H})$,

$$u^* = \min_{u \in \mathcal{H}} E(u) + \frac{1}{2}||u - f||^2, \tag{14.3}$$

admits a unique solution u^*, and we set

$$\text{prox}_E(f) := u^*. \tag{14.4}$$

The operator $\text{prox}_E : \mathcal{H} \to \mathcal{H}$ is called proximity operator because of the interpretation of (14.3) as a 'relaxed' projection problem.

14.1 The Proximal Point Algorithm

We generalise (14.3) by the following generic formulation.

Problem 14.1 *Let E_1 and E_2 be two functions in $\Gamma_0(\mathcal{H})$ such that E_2 has a bounded subgradient. The objective is to minimise $E = E_1 + E_2$ over $\mathcal{H}$.*

One can prove

Theorem 14.1 *Let $u \in \mathcal{H}$ and $\gamma \in]0, +\infty[$. Then u solves Problem 14.1 if and only if*

$$u = \text{prox}_{\gamma E_1}(u - \gamma \nabla E_2(u)). \tag{14.5}$$

The formula (14.5) inspires a convergent fixed point algorithm. Because of the use of the proximity operator $\text{prox}_{\gamma E_1}$ the resulting method is called proximal point algorithm.

Remark

Although the task is to minimise $E_1 + E_2$, in (14.5) we effectively minimise only w.r.t. E_1, while a gradient descent scheme for minimising E_2 is encoded in the argument in (14.5).

Can we approach the problem in a more general way?

14.2 Monotone Operators

An operator T on a Hilbert space $\mathcal{H}$ is a 'point-to-set' mapping

$$T : \mathcal{H} \to 2^{\mathcal{H}}. \tag{14.6}$$

Thereby, $2^{\mathcal{H}}$ denotes the power set of $\mathcal{H}$, i.e. the set of all possible subsets of $\mathcal{H}$.

We will make no distinction between T and its graph, i.e. the set $\{(x, y) : y \in T(x)\}$. Thus, we may simply say that an operator is any subset T of $\mathcal{H} \times \mathcal{H}$, and define $T(x) := Tx := \{y : (x, y) \in T\}$.

Definition 14.2 *Let $(x, y) \in T$.*

a) The domain of T is its 'projection' onto the first coordinate: dom $T = x$.

b) The range (or image) of T is its 'projection' onto the second coordinate: im $T = y$.

c) The inverse T^{-1} of T is in (y, x).

d) I denotes the identity operator (x, x).

Two basic constructions are as follows.

Definition 14.3 *Let A, B be two operators, and $c \in \mathbb{R}$. Then*

$$cT := \{(x, cy) : (x, y) \in T\} \tag{14.7}$$

$$A + B := \{(x, y + z) : (x, y) \in A, (x, z) \in B\}. \tag{14.8}$$

An operator is monotone, if

$$\langle x' - x, y' - y \rangle \geq 0 \qquad \forall\, (x, y)\,, \left(x', y'\right) \in T. \tag{14.9}$$

A monotone operator is maximal if — considered as a graph — it is not strictly contained in any other monotone operator on $\mathcal{H}$.

One can show:

Theorem 14.2 *If E is a proper convex and LSC function, then the subgradient ∂E is a maximal monotone operator.*

While Theorem 14.2 is useful for deriving a basic algorithm, the following assertion is useful for constructing splittings.

Theorem 14.3 *T is maximal monotone if and only if, for any $x \in X$ and any scalar $\lambda > 0$, there exists a unique $z \in X$ such that $x \in (I + \lambda T)(z)$.*

By Theorem 14.3, for any maximal monotone mapping T, the mapping

$$J_T^{\lambda} := (I + \lambda T)^{-1}, \tag{14.10}$$

called the resolvent of T, is a single-valued mapping defined everywhere on X.

A particular consequence of Theorem 14.3 is

$$0 \in T(x) \Leftrightarrow x = J_T^{\lambda}(x). \tag{14.11}$$

14.3 Problem Structure Revisited

Let us write the considered optimisation task as

$$\min_{u \in \mathcal{H}} F(u) \tag{14.12}$$

for F proper convex and LSC. Then the first order optimality condition, in the described context sometimes called Fermat's rule, reads as

$$0 \in \partial F(u), \tag{14.13}$$

where we write "$\in$" as ∂F is set-valued by definition.

By Theorem 14.2, ∂F is a maximal monotone operator. Then the resolvent $J^{\lambda}_{\partial F}$ as by (14.10) is defined.

The proximal point algorithm for solving the multi-valued equation (14.13) relies on (14.11). It generates a sequence $u^{(k)}$ by the recurrence

$$u^{(k+1)} = J^{\lambda}_{\partial F}\left(u^{(k)}\right). \tag{14.14}$$

This method is known to converge to the solution of (14.12) from an arbitrary $u^{(0)}$.

In some cases — e.g. in some (not all!) settings in which F is composed of a data term and a smoothness term — F can be expressed as a sum of two proper convex and LSC functions, $F := R + S$, in such a way that the resolvents $J^{\lambda}_{\partial R}$ and $J^{\lambda}_{\partial S}$ are much easier to evaluate than $J^{\lambda}_{\partial F}$.

Splitting algorithms are designed to solve

$$0 \in \partial F(u) = \partial R(u) + \partial S(U), \tag{14.15}$$

using $J^{\lambda}_{\partial R}$ and $J^{\lambda}_{\partial S}$ instead of $J^{\lambda}_{\partial F}$.

The inclusion (14.15) can be rewritten as:

$$\begin{aligned}
\partial S(u) \in -\partial R(u) \quad &\Leftrightarrow \quad u + \eta \partial S(u) \in u - \eta \partial R(u), && \eta > 0 \\
&\Leftrightarrow \quad (I + \eta \partial S)(u) \in (I - \eta \partial R)(u), && \eta > 0 \\
&\Leftrightarrow \quad u \in J^{\lambda}_{\partial S}(I - \eta \partial R)(u), && \eta > 0. \qquad (14.16)
\end{aligned}$$

The corresponding fixed point iteration

$$u^{(k+1)} = J^{\eta}_{\partial S}(I - \eta \partial R)\left(u^{(k)}\right) \tag{14.17}$$

is called forward-backward-splitting.

Remark

One may also rewrite (14.15) as $\partial R\left(u\right) \in -\partial S\left(u\right)$.

The Douglas-Rachford splitting algorithm appears to allow the most attractive mathematical convergence properties of splitting schemes. It reads as

$$u^{(k+1)} = \left[J^{\lambda}_{\partial R}\left(2J^{\lambda}_{\partial S} - I\right) + \left(I - J^{\lambda}_{\partial S}\right)\right]\left(u^{(k)}\right). \tag{14.18}$$

Summary

- Splitting algorithms rely on proper convex and LSC energy functionals.
- Splitting algorithms rely on an additive decomposition of a given energy functional.

15 Fast Optimisation and Rank-Deficient Problems

Let us consider the problem

$$\min_{\vec{x}\in\mathbb{R}^n} \|F(\vec{x})\|_2^2, \qquad F:\mathbb{R}^n \to \mathbb{R}^n \tag{15.1}$$

Aiming for a fast solver, we substitute in a first step F with its Taylor linearisation:

$$\min_{\vec{x}^{k+1}\in\mathbb{R}^n} \|F(\vec{x}^k) + F'(\vec{x}^k)\cdot(\vec{x}^{k+1} - \vec{x}^k)\|_2^2 \tag{15.2}$$

where F' denotes the $n \times n$ Jacobi matrix and $\vec{x}^k$ an iterate; the solution of (15.2) gives the next iterate $\vec{x}^{k+1}$. Defining

$$A := F'(\vec{x}), \quad x := \vec{x}^{k+1} - \vec{x}^k, \quad b := -F(\vec{x}^k) \tag{15.3}$$

we notice that the problem is equivalent to minimising the residual

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 \tag{15.4}$$

An efficient numerical treatment relies on the use of orthogonal matrices.

Def.: A matrix Q is orthogonal, if and only if

(i) its columns are pairwise orthogonal

(ii) $\|Qx\|_2 = \|x\|_2$.

One can show that it holds $Q^\top = Q^{-1}$, and of course $Q^\top$ is also orthogonal.

Introducing a clever choice of $Q^\top$ into (15.4), the idea is that we solve instead a simpler problem

$$\min_{\vec{x}} \|Q^\top(Ax - b)\|_2^2 \tag{15.5}$$

The numerical approach relies on the QR-decomposition of A:

Theorem: There is an orthogonal matrix Q such that $A = QR$ where R is upper triangular with non-negative diagonal elements.

The factorisation $A = QR$ is the key to define in a general setting the obvious iteration

$$\vec{x}^{k+1} := \vec{x}^k - F'(\vec{x}^k)^\dagger \cdot F(\vec{x}^k) \tag{15.6}$$

where $F'(\vec{x}^k)^\dagger$ is a generalised (pseudo-) iverse:

(a) The setting can be extended to $F : \mathbb{R}^n \to \mathbb{R}^m$ for $m \geq n$ (related to least-squares-approximation)

(b) $F'(\vec{x}^k)$ can be rank-deficient

(c) One can couple (15.6) with a trust-region idea.

For (a) and/or (b) the method is called *Gauß-Newton-scheme*, adding the option (c) yields a *Levenberg-Marquardt-scheme*.

15.1 Theory of QR-Decomposition

We now introduce the *augmented system*:

Theorem: Assume that $A \in \mathbb{R}^{n\times n}$ has rank n. Then the symmetric linear system

$$\underbrace{\begin{pmatrix} I & A \\ A^\top & 0 \end{pmatrix}}_{\in\mathbb{R}^{2n\times 2n}} \underbrace{\begin{pmatrix} y \\ x \end{pmatrix}}_{\in\mathbb{R}^{2n}} = \underbrace{\begin{pmatrix} y \\ x \end{pmatrix}}_{\in\mathbb{R}^{2n}} \tag{15.7}$$

is nonsingular and solves $\min_x \|Ax - b\|_2^2$, where y takes the role of the residual $y = b - Ax$.

We now show how to use the QR-decomposition to solve (15.7).

Theorem: Let $\mathrm{rank}(A) = n$ and $A = QR$. Then the solution of (15.7) can be computed from

$$d := Q^\top b, \qquad x := R^{-1}d \tag{15.8}$$

Proof: The augmented system can be written as

$$y + Ax = b, \qquad A^\top y = 0. \tag{15.9}$$

Using $A = QR$ we obtain

$$y + QRx = b, \qquad R^\top Q^\top y = 0. \tag{15.10}$$

Multiplying the first equation with $Q^\top$ and the second with $(R^\top)^{-1} =: R^{-\top}$, we get

$$Q^\top y + Rx = Q^\top b, \qquad Q^\top y = 0. \tag{15.11}$$

Using the second equation to eliminate the first summand in the first equation, we can solve for x via $x = R^{-1}Q^\top b = R^{-1}d$.

□.

We now show how to modify the QR-decomposition for the rank-deficient case.

Theorem: Let $A \in \mathbb{R}^{n\times n}$ with $\mathrm{rank}(A) := r < n$, then there is a permutation matrix Π such that

$$A\Pi \;=\; Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{matrix} \}\, r \\ \}\, n-r \end{matrix} \tag{15.12}$$

where $R_{11} \in \mathbb{R}^{r\times r}$ is upper triangular with positive diagonal elements.

The *proof* relies on sorting the columns of A such that $A\Pi = (A_1, A_2)$ where $A_1 \in \mathbb{R}^{n\times r}$ has r linearly independent columns. Remember in this case not to forget to sort the unknowns in x plus the entries of b in accordance.

15.2 Computing Q and R

We consider here the method of Gram and Schmidt (GS). Its basic idea is to produce the columns of Q successively by orthogonalising the columns of A.
Assume $q_1, \ldots, q_k$ have been determined, and the q_i have been normalised. Then, we can compute the *projections*

$$r_{ik} \;=\; q_i \cdot a_k \tag{15.13}$$

for $i = 1, \ldots, k-1$. This determines the amount of the column vector a_k lying in the space spanned by $\{q_1, \ldots, q_{k-1}\}$. The remaining part of a_k is

$$\hat{q} \;=\; a_k - \sum_{i=1}^{k-1} r_{ik} q_i \tag{15.14}$$

which still needs to be normalised. Successive application leads to the GS algorithm, where Q and R are generated columnwise.

A *modified version* (MGS) of this method is obtained by subtracting the projections $r_{ik}q_i$ from a_k as soon as these are computed, i.e.

$$a_j^{(k+1)} := a_j^{(k)} - r_{kj} q_k, \qquad r_{kj} := q_k \cdot a_j^{(k)}, \tag{15.15}$$

for $j = k+1, \ldots, n$. The MGS method is known to be more stable and accurate.