Introduction to Variational Methods

for

Computer Vision and Image Processing

Coloma Ballester

coloma.ballester@upf.edu
Universitat Pompeu Fabra

Slides credits: J.F. Garamendi

Optimization and Inference Techniques for Computer Vision

Master in
Computer Vision
Barcelona

Outline

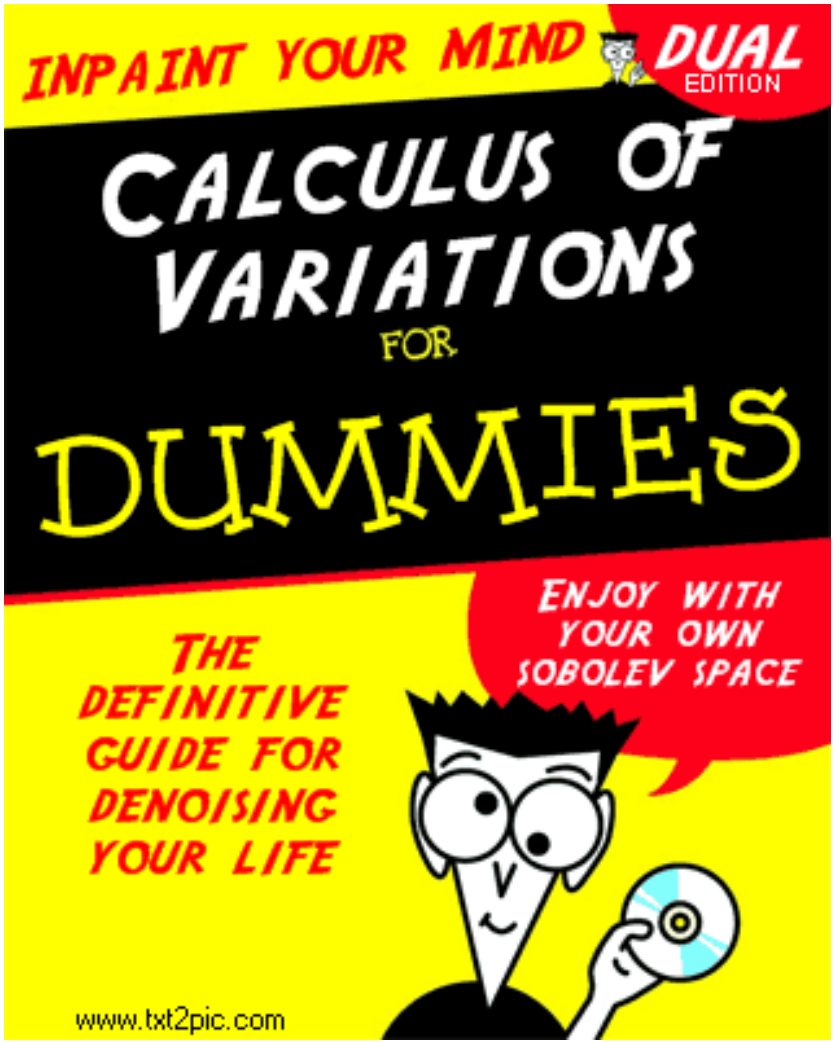
1. Some basic ideas and definitions.
2. A little bit of review on Differential calculus
   - Gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$.
   - Level sets (and level lines) of a function $f : \mathbb{R}^n \to \mathbb{R}$, and its geometrical relation to the gradient.
   - Directional derivatives, and how to compute them using the gradient, ...
3. Introduction to optimization problems and energy minimization methods
4. Gateaux derivative
5. Euler-Lagrange equation and gradient methods
6. Examples

Introduction to Variational methods

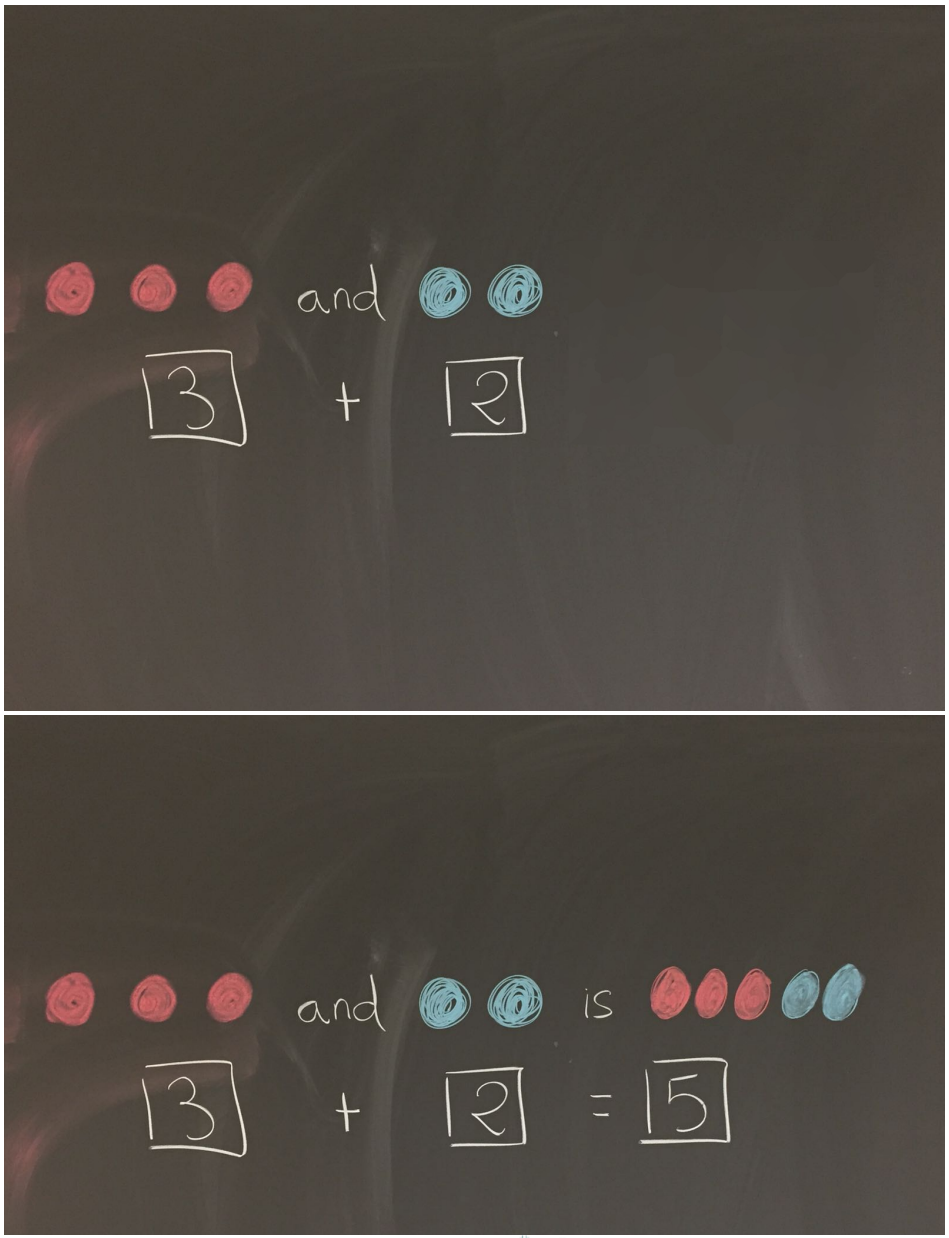
for Computer Vision and Image Processing

This is a conceptual (and not formal) introduction to understand

- Relationship between digital image processing, mathematical analysis and linear algebra.
- Problem modelling as energy functional minimization.
- Partial differential equations (PDEs) and why they appear.
- Algebraic system of equations and why they appear.
- Notation.



Mathematical Modelling



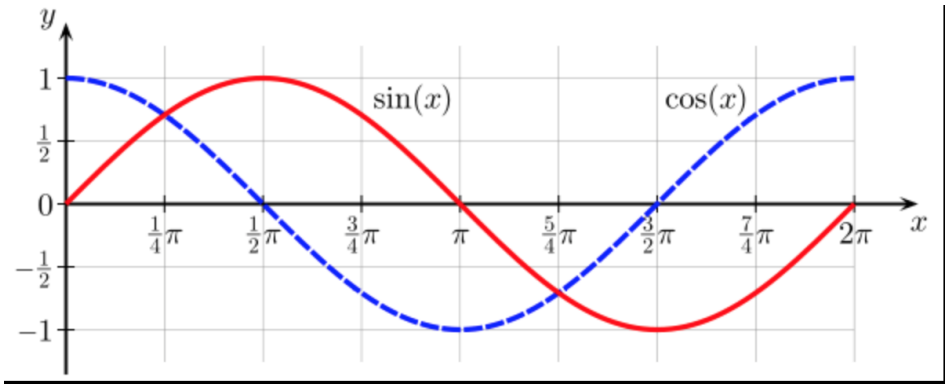
- A mathematical model is a description of a system using mathematical concepts and language.
- A Model may help to explain a system and to study the effects of different components, and to make

A (mathematical) concept we will need

Function

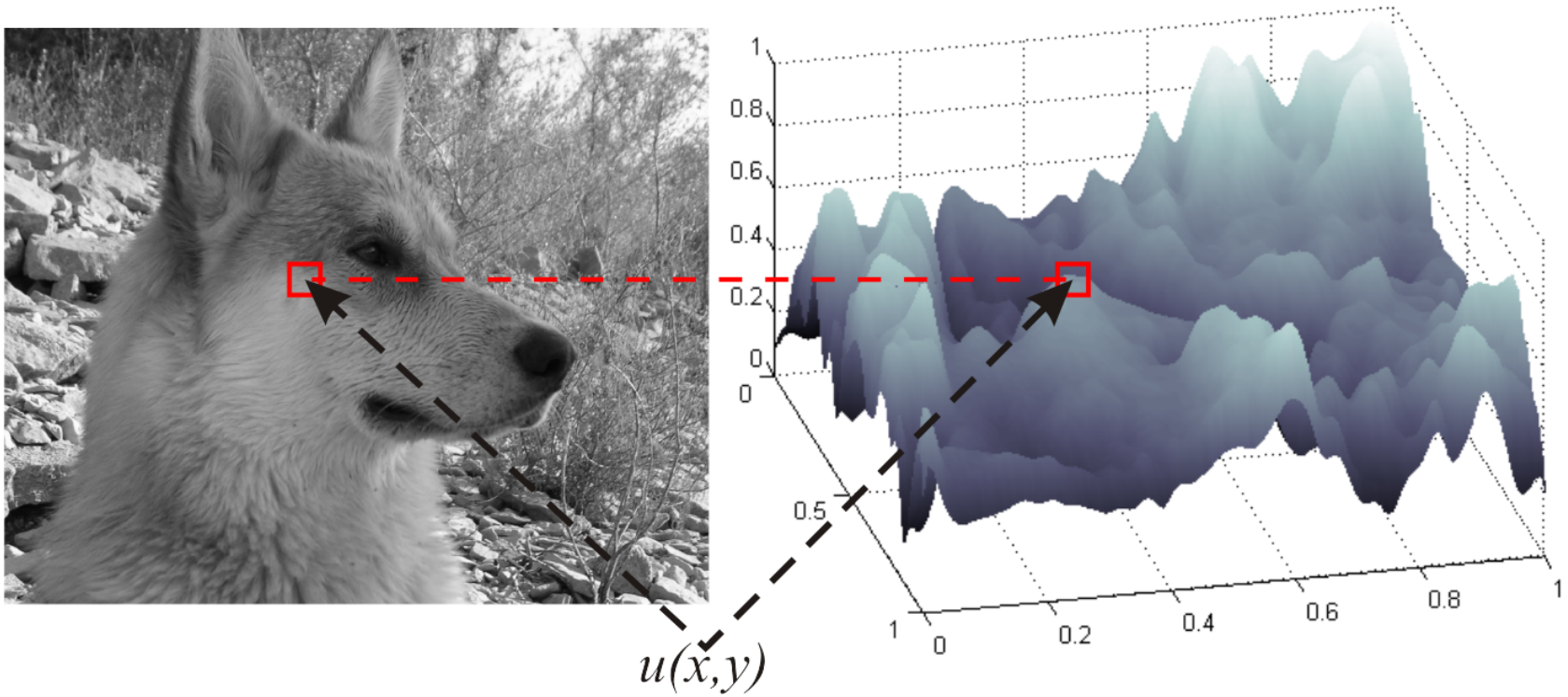
- Relation between a set of inputs and a set of permissible outputs.
- Each input is related to exactly one output.
- There are many ways to describe or represent a function
  - By a formula. $f(x) = x^2$
  - By an algorithm that tells how to compute the output.
  - **By a picture, called the graph of a function.**
  - By a table,
  - Implicitly as a solution of a (partial) differential equation.

(Wikipedia)



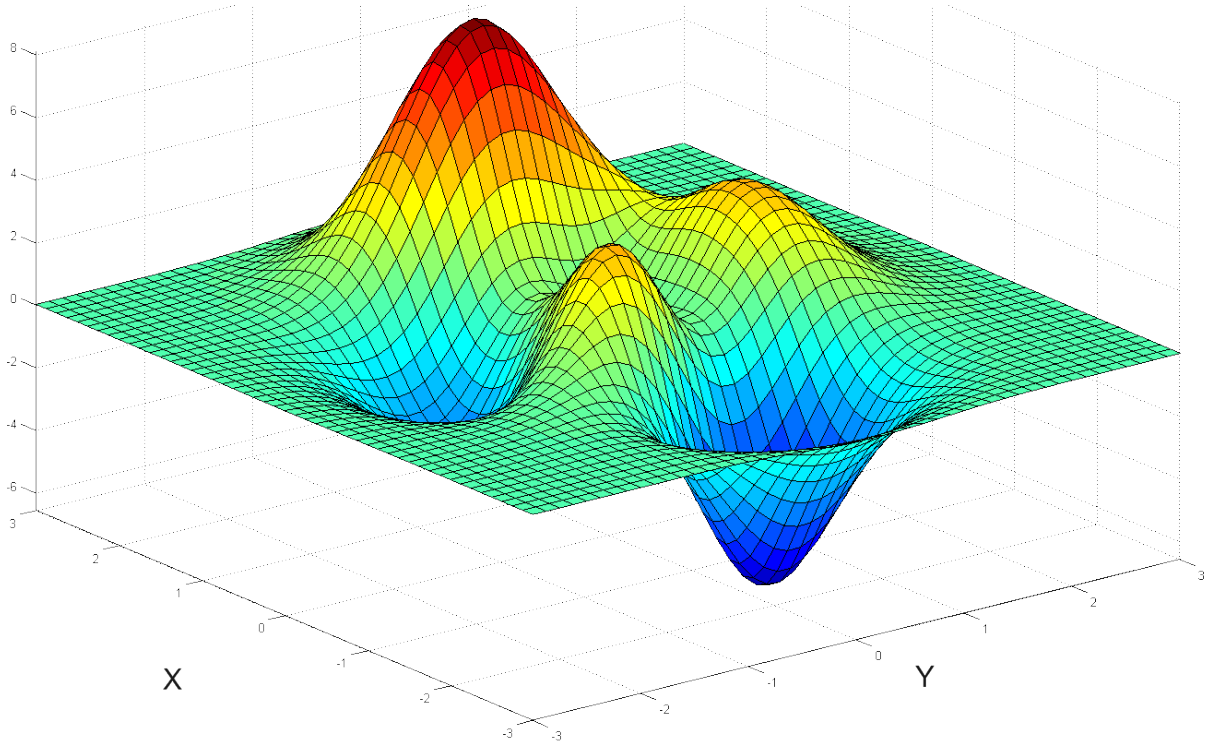
Modelling Images as Functions

For each position (x, y) there is a value $u(x, y)$.



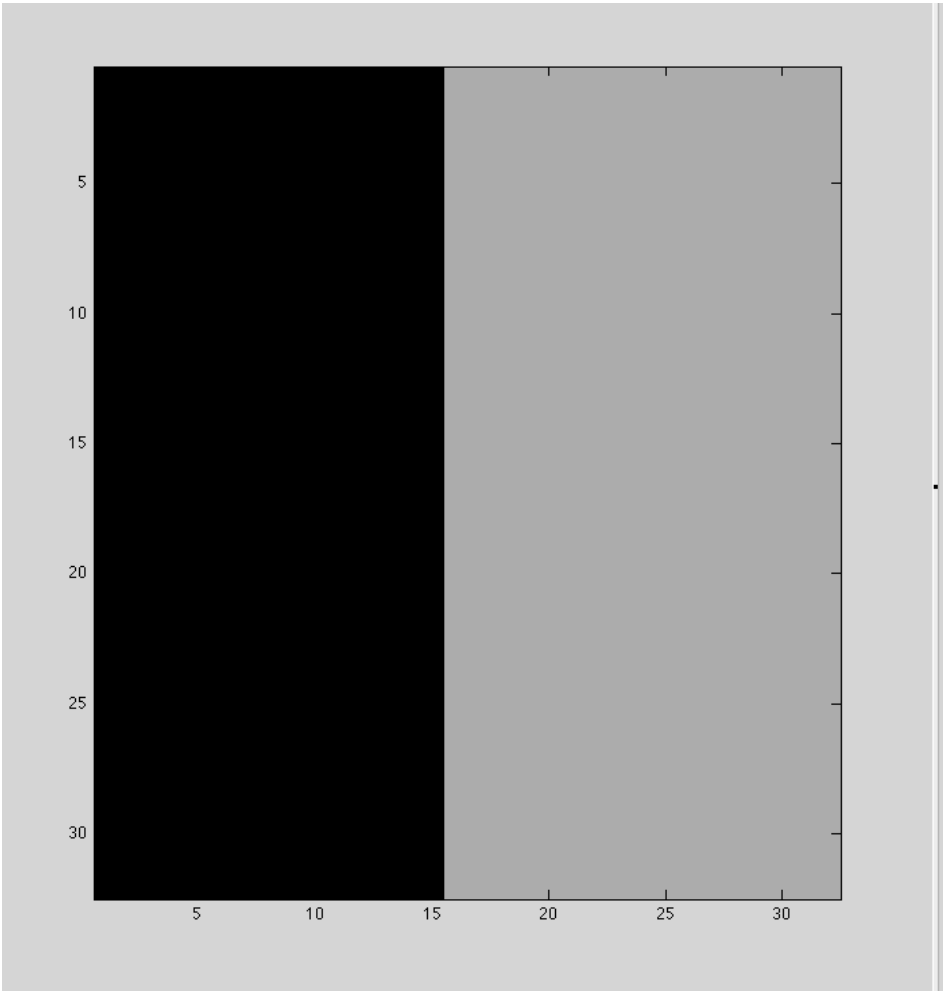
- Image u(x,y) with $u : \Omega \rightarrow \mathbb{R}$
- $\Omega \subset \mathbb{R}^2$, $\Omega = [0, 1]$x$[0, 1]$
- $(x, y) \in \Omega$ denotes pixel location

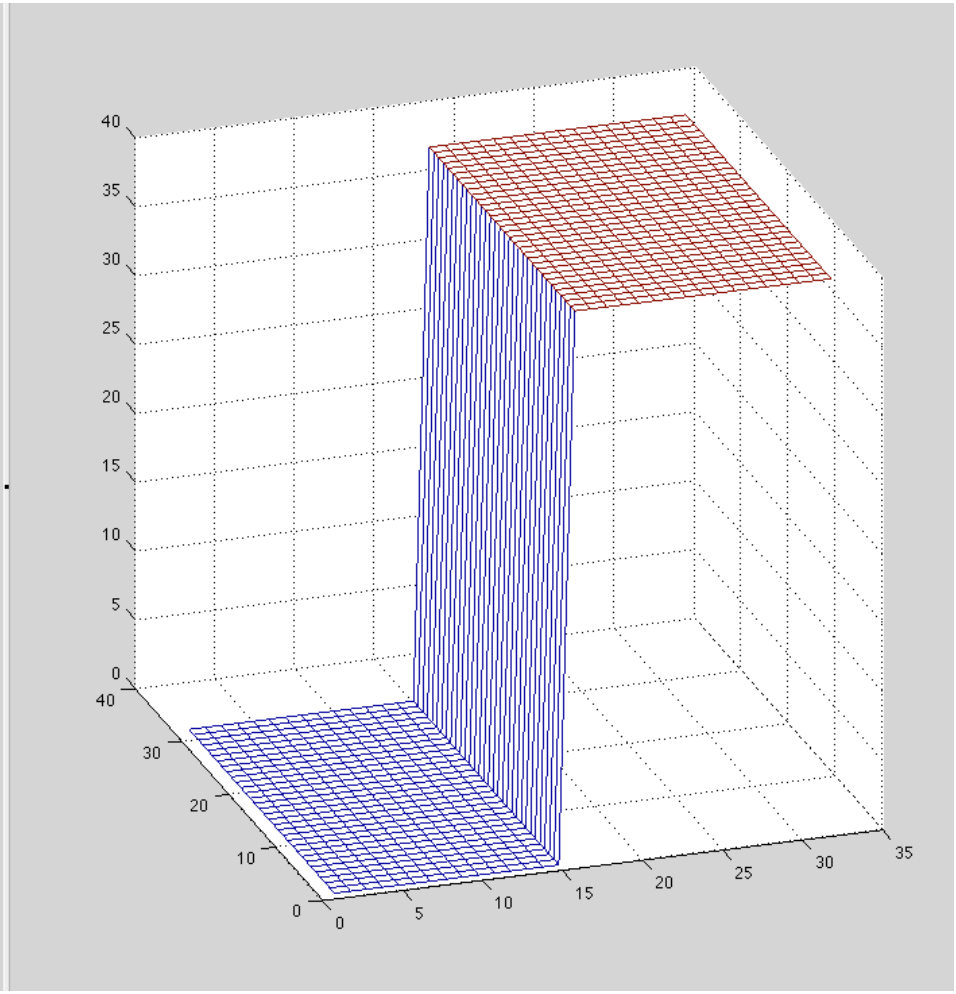
Modelling Images as Functions

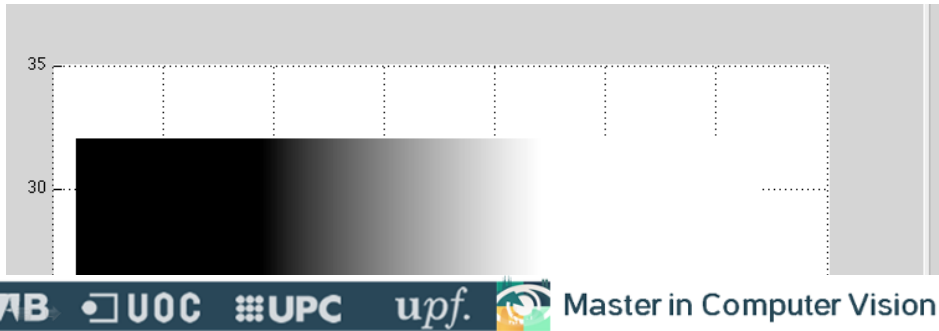


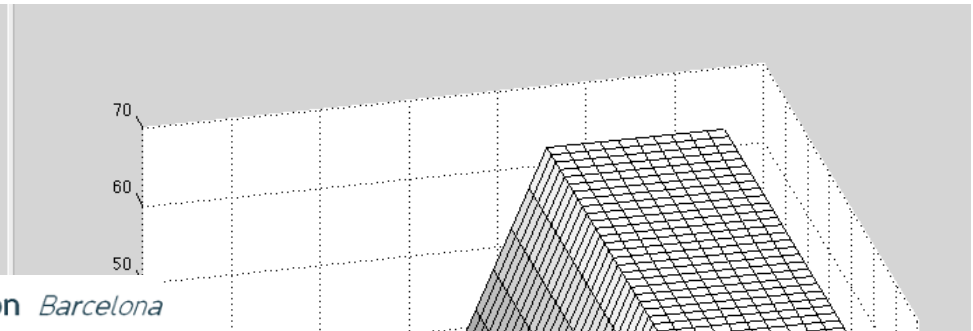
Now forget where you come from and apply everything you know about functions

Modelling Images as Functions

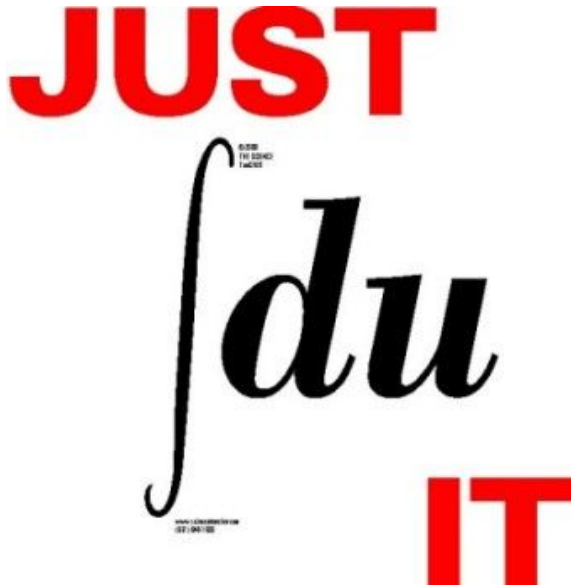








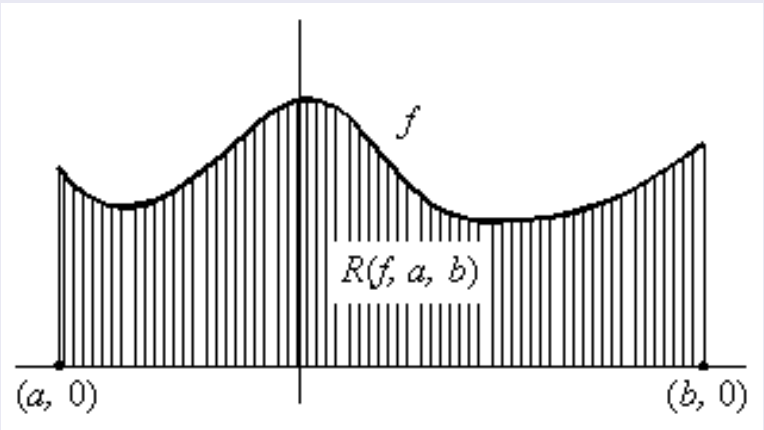
What can we do with functions?



What can we do with functions?

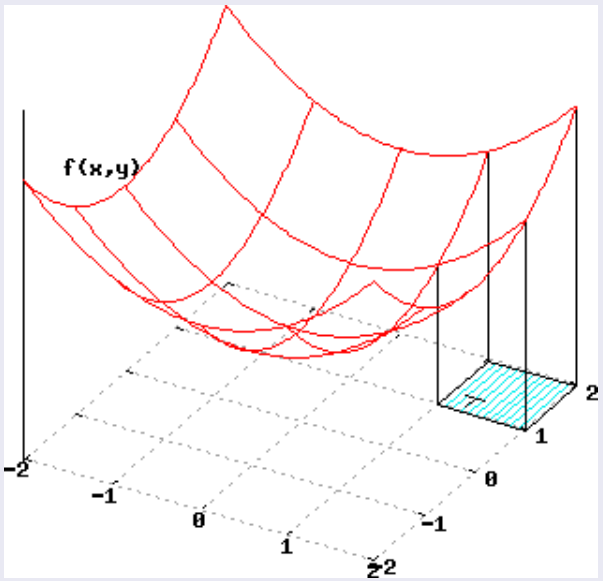
Integral

f(x)



$$\int_a^b f(x)dx$$

f(x,y)

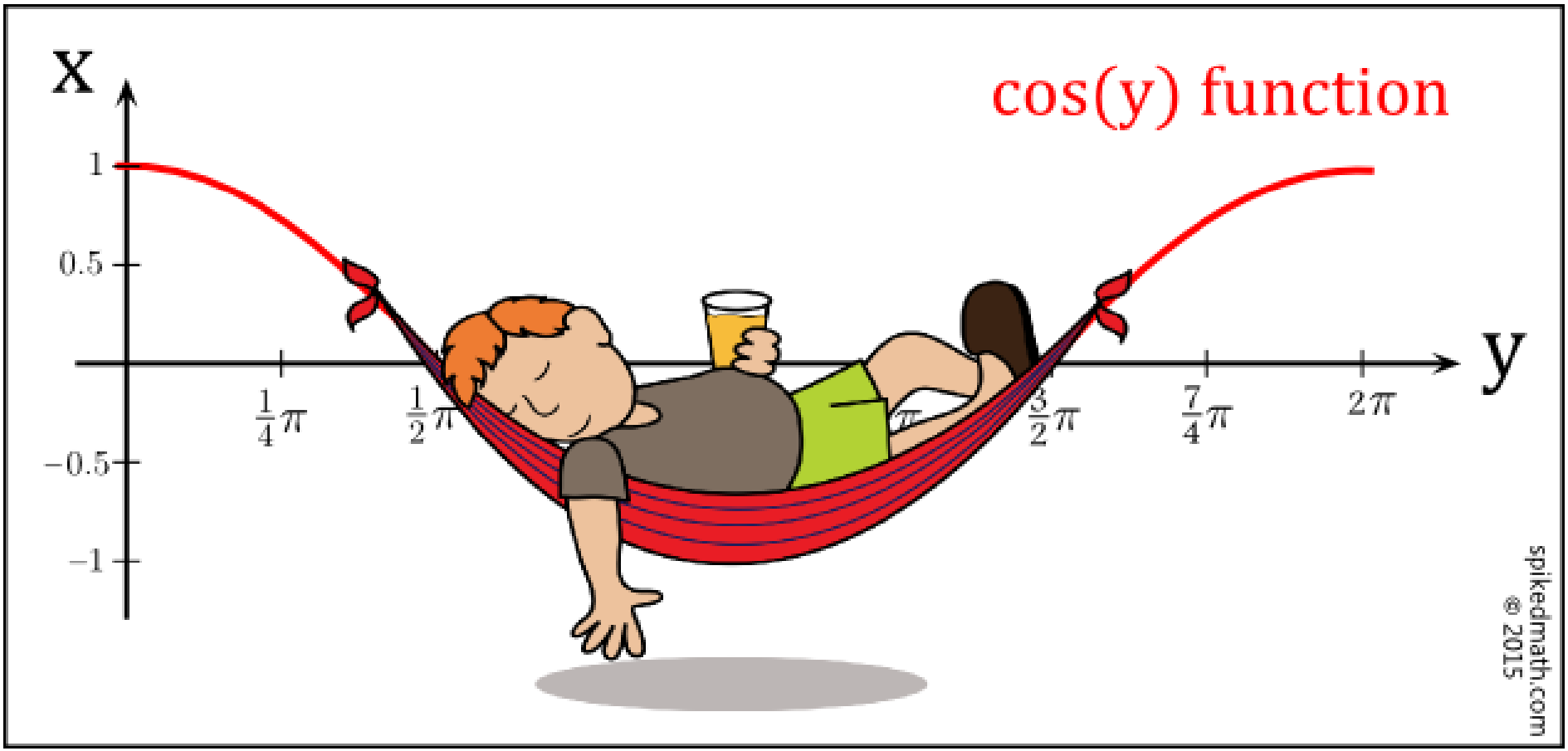


$$\int_a^b \int_c^d f(x,y)dxdy$$

$$\int_\Omega f(x,y)dxdy$$

The integral sign $\int$ represents integration. Integral: The area under the

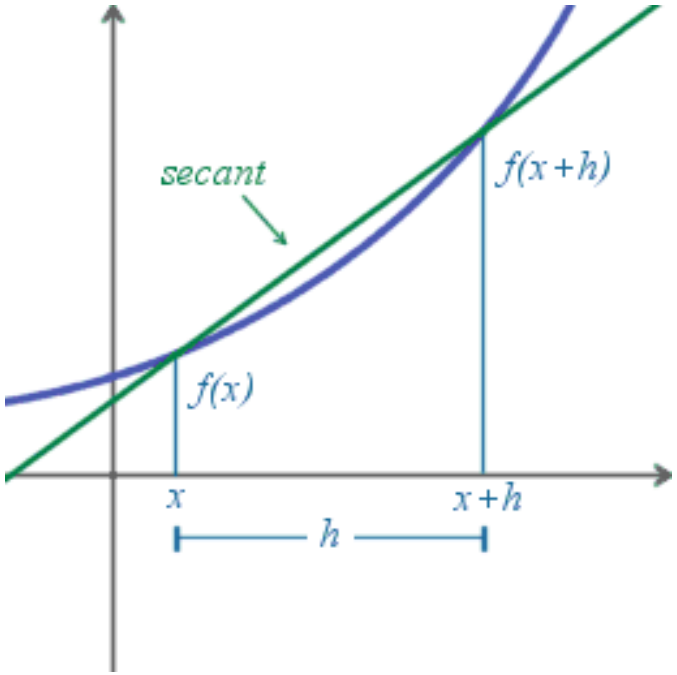
What can we do with functions?



What can we do with functions?

Derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



The derivative is a measure of how a function (locally) **changes** as its input changes by a small amount h. Given a function, the result is another function. It is a LOCAL measure.

What can we do with functions?

The 1D case: Derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Let's be h very, very small, very close to 0, then

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

$$hf'(x) \approx f(x+h) - f(x)$$

$$f(x+h) \approx f(x) + hf'(x)$$

Notice the last expression is the first order Taylor's approximation of $f(x+h)$ at point x. Keep this in mind, you will need for the **Optical Flow** problem.

What can we do with functions?

The 2D case: Directional Derivative

Suppose that $f : \mathbb{R}^2 \to \mathbb{R}$. Now we have infinite directions $\vec{v}$ over we can measure the rate of change, just choose one given by the unitary vector $\vec{v}$, then

$$D_{\vec{v}} f(\vec{X}) = \lim_{\epsilon \to 0} \frac{f(\vec{X} + \epsilon \vec{v}) - f(\vec{X})}{\epsilon}$$

where $\vec{X} = (x_1, x_2)$.
Analogy with the 1D case: $\vec{v} = \epsilon \vec{v} \longrightarrow_{\epsilon \to 0} 0$.

Special case: $\vec{v}$ is a vector of the basis, then we call it **Partial Derivative**

What can we do with functions?

The 2D case: Gradient

Partial derivatives

For $\vec{v} = (1, 0)$

$$\frac{\partial f}{\partial x_1}(x_1, x_2) = \lim_{\epsilon \to 0} \frac{f(x_1 + \epsilon, x_2) - f(x_1, x_2)}{\epsilon}$$

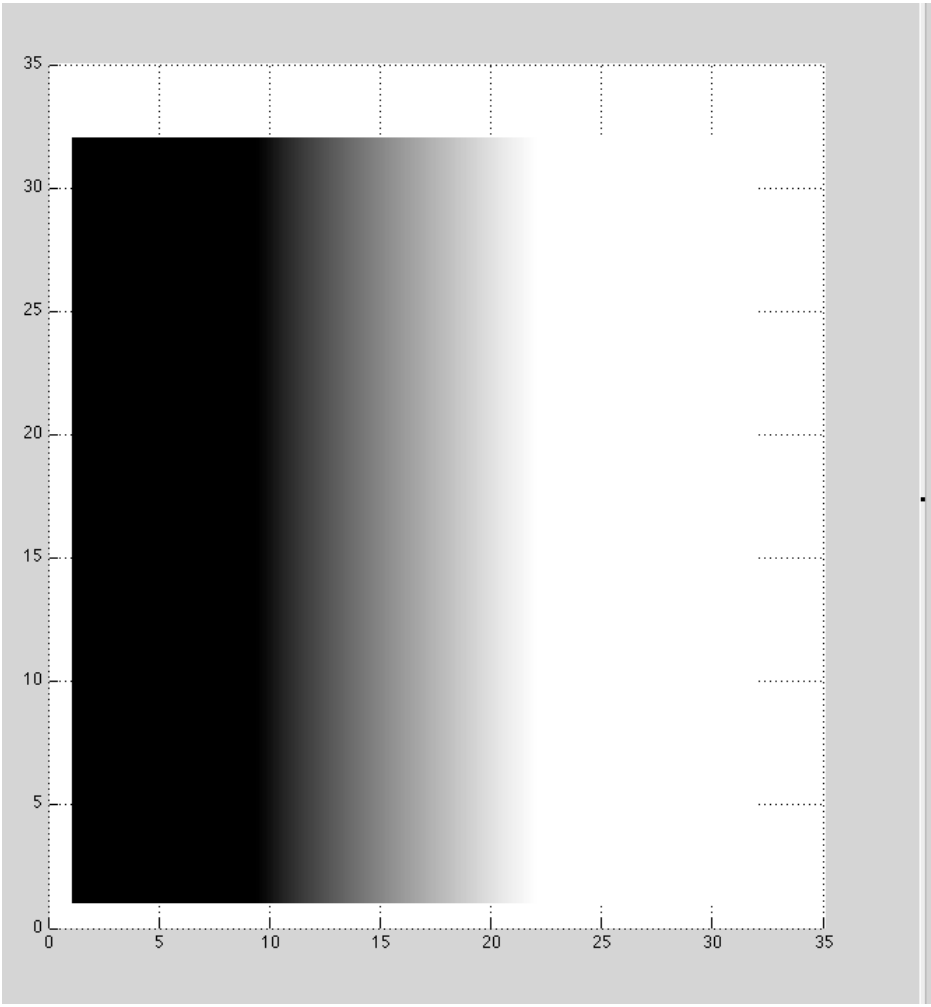
For $\vec{v} = (0, 1)$

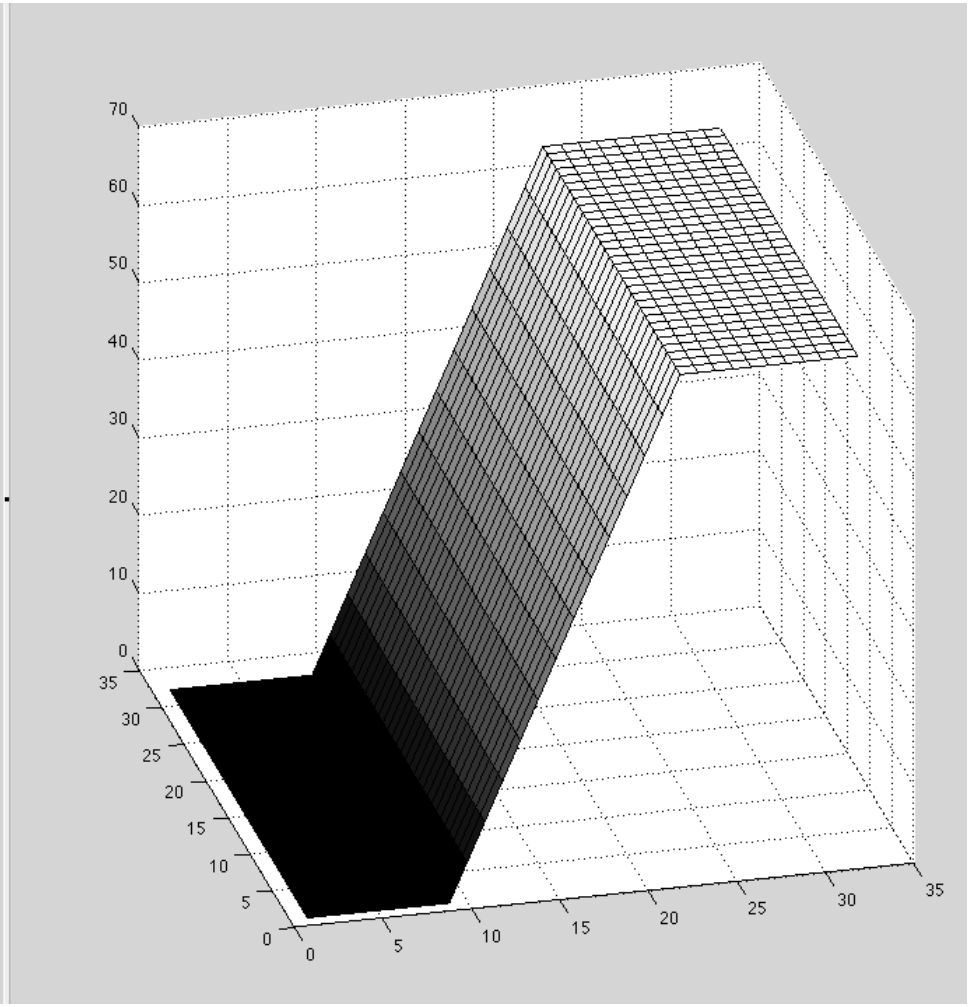
$$\frac{\partial f}{\partial x_2}(x_1, x_2) = \lim_{\epsilon \to 0} \frac{f(x_1, x_2 + \epsilon) - f(x_1, x_2)}{\epsilon}$$

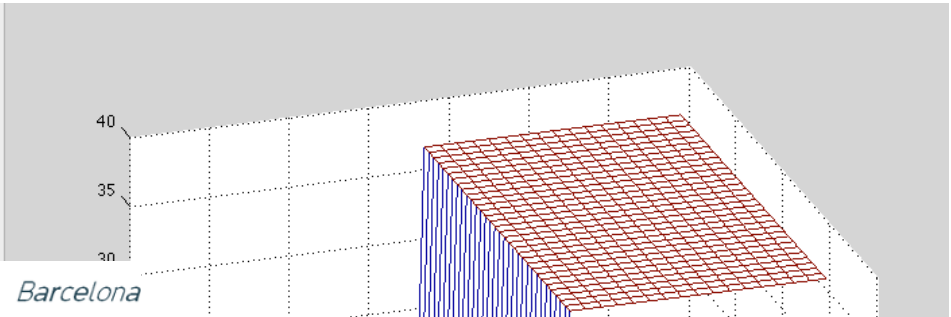
Gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

Examples







What can we do with functions?

The 2D case: Gradient

Partial derivatives Gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

Relationship between Gradient and Directional derivative

$$D_{\vec{v}} f(\vec{X}) = < \nabla f, \vec{v} >$$

for all direction $\vec{v} \in \mathbb{R}^n$.

Meaning of the gradient

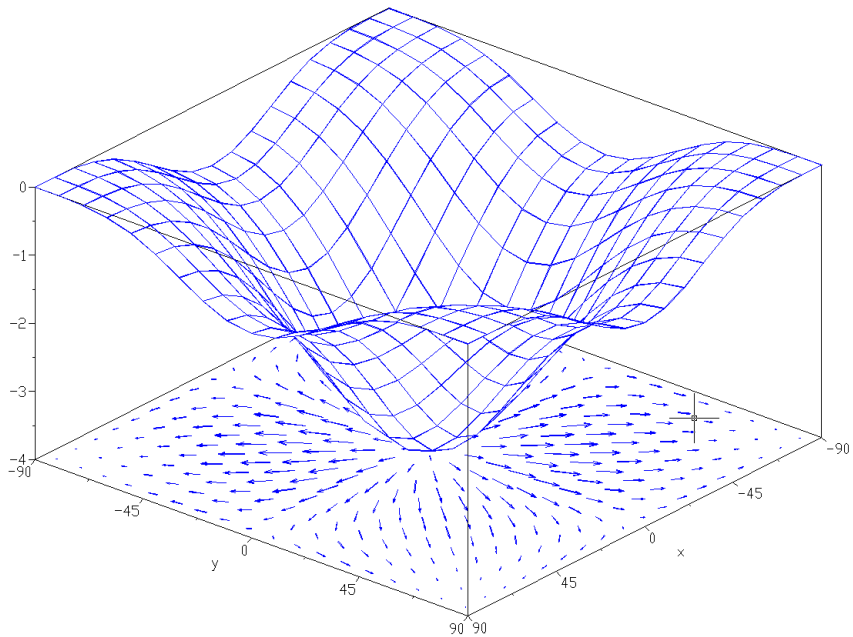
The gradient is a vector that points out in the direction where the function increases more rapidly.

$f : O \to \mathbb{R}$, $O \subset \mathbb{R}^N$ open set.
$f(x)$ represents the height of the topography at the point $x \in O$.
Thus, the graph of f, that is, the set

$$\text{Graph}(f) := \{(x, f(x)) : x \in O\}.$$

The gradient of f at x_0, that is the vector $\nabla f(x_0)$ points in the direction where the mountain is more steep.

Thus $-\nabla \mathbf{f}(\mathbf{x_0})$ points to the direction of **steepest descent** (the fastest descent).



Meaning of the gradient

Indeed:

The **directional derivative of f at the point x in the direction v** $\in \mathbb{R}^{\mathbf{N}}$ is the rate of variation of f at the point x when we perturb it with v, that is

$$\mathbf{D_v f(x)} = \lim_{\epsilon \to 0+} \frac{f(\tilde{x} + \epsilon v) - f(\tilde{x})}{\epsilon} \left(= \frac{d}{dt} f(\tilde{x} + tv)|_{t=0} \right),$$

if the limit exists. If it exists we denote this limit by $D_v f(\tilde{x})$.
If f admits a derivative at the point $\tilde{x}$, then f admits a directional derivative at $\tilde{x}$ in any direction v and the following formula holds

$$D_v f(\tilde{x}) = \langle \nabla f(\tilde{x}), v \rangle. \tag{1}$$

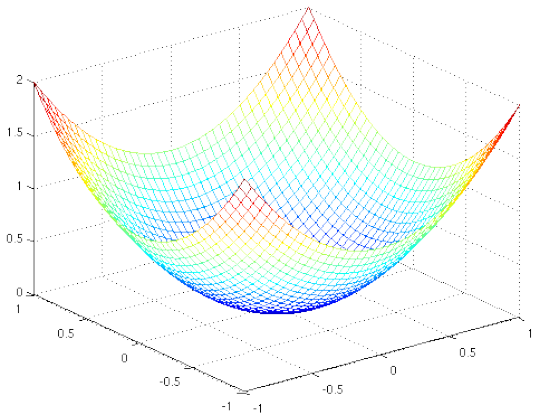
The gradient permits to express the directional derivatives.

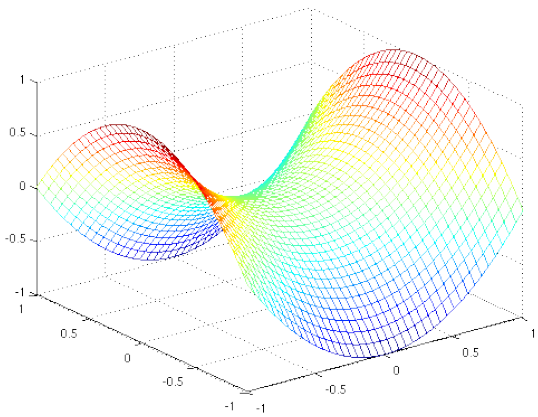
Which vector $v \in \mathbb{R}^N$ of norm 1 would you take to maximize the right hand side of (1)?
Which vector $v \in \mathbb{R}^N$ of norm 1 would you take to minimize the right hand side of (1)?

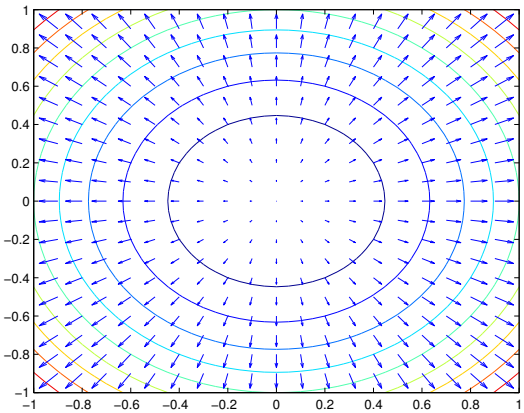
Meaning of the gradient

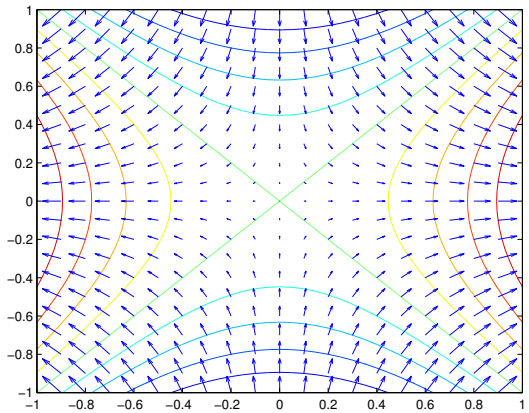
Examples

The gradient ∇f points in the direction of steepest ascent:









Meaning of the gradient

Let $c \in \mathbb{R}$ and $S_c(f) = \{x \in O : f(x) = c\}$. $S_c(f)$ is called the **level set of f associated to the level c**.

To simplify, assume $N = 2$. Then, $\mathrm{Graph}(f)$ is a surface in $\mathbb{R}^3$ and, for c fixed, $S_c(f)$ is a curve in $\mathbb{R}^2$, which is called the **level line** associated to c.

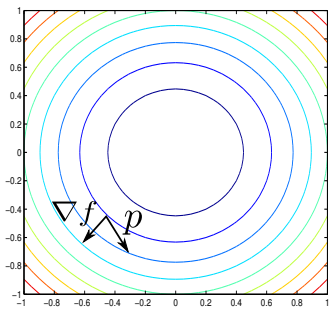
Consider a parametrization $\gamma : [0, 1] \to S_c(f)$ of the curve $S_c(f)$. Thus, at the points $\gamma(t) \in S_c(f) \subset \mathbb{R}^2$, we have

$$f(\gamma(t)) = c \quad (\text{i.e., constant}) \; \forall t.$$

Then, by the chain rule,

$$\langle \nabla f(\gamma(t)), \gamma'(t) \rangle = 0,$$

for all t, where $\gamma'(t)$ denotes the tangent vector to the level line, at $\gamma(t)$.
We have obtained that the gradient is always perpendicular to the level line.



More on the short reminder

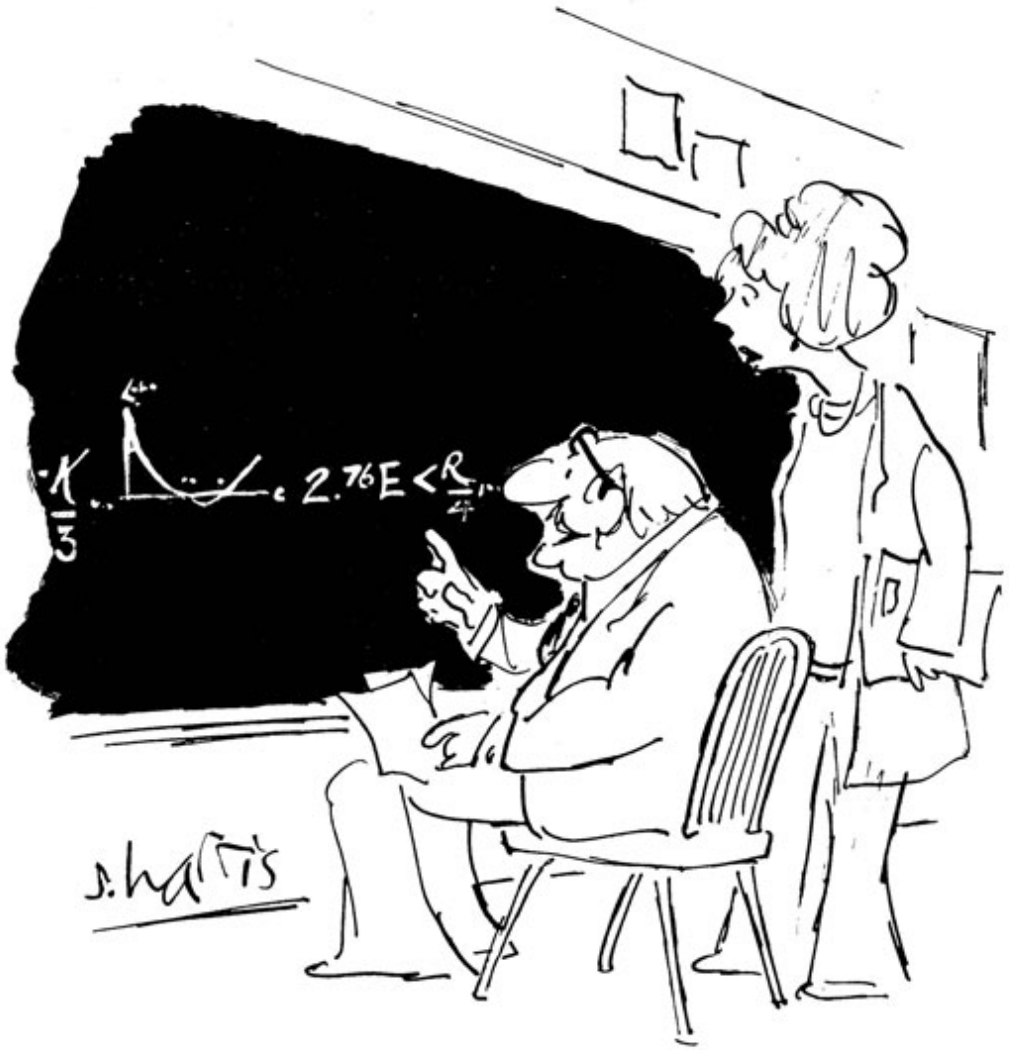
Gradient Module

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2}$$

Laplacian

$$\Delta f = (\nabla \cdot \nabla) f = \nabla^2 f = \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2}$$

On Images


"THE BEAUTY OF THIS IS THAT IT IS ONLY OF THEORETICAL IMPORTANCE, AND THERE IS NO WAY IT CAN BE OF ANY PRACTICAL USE WHATSOEVER."

What can we do with functions?

Example of Integral

$$\int_{\Omega} |u(x,y) - f(x,y)|^2 dxdy$$

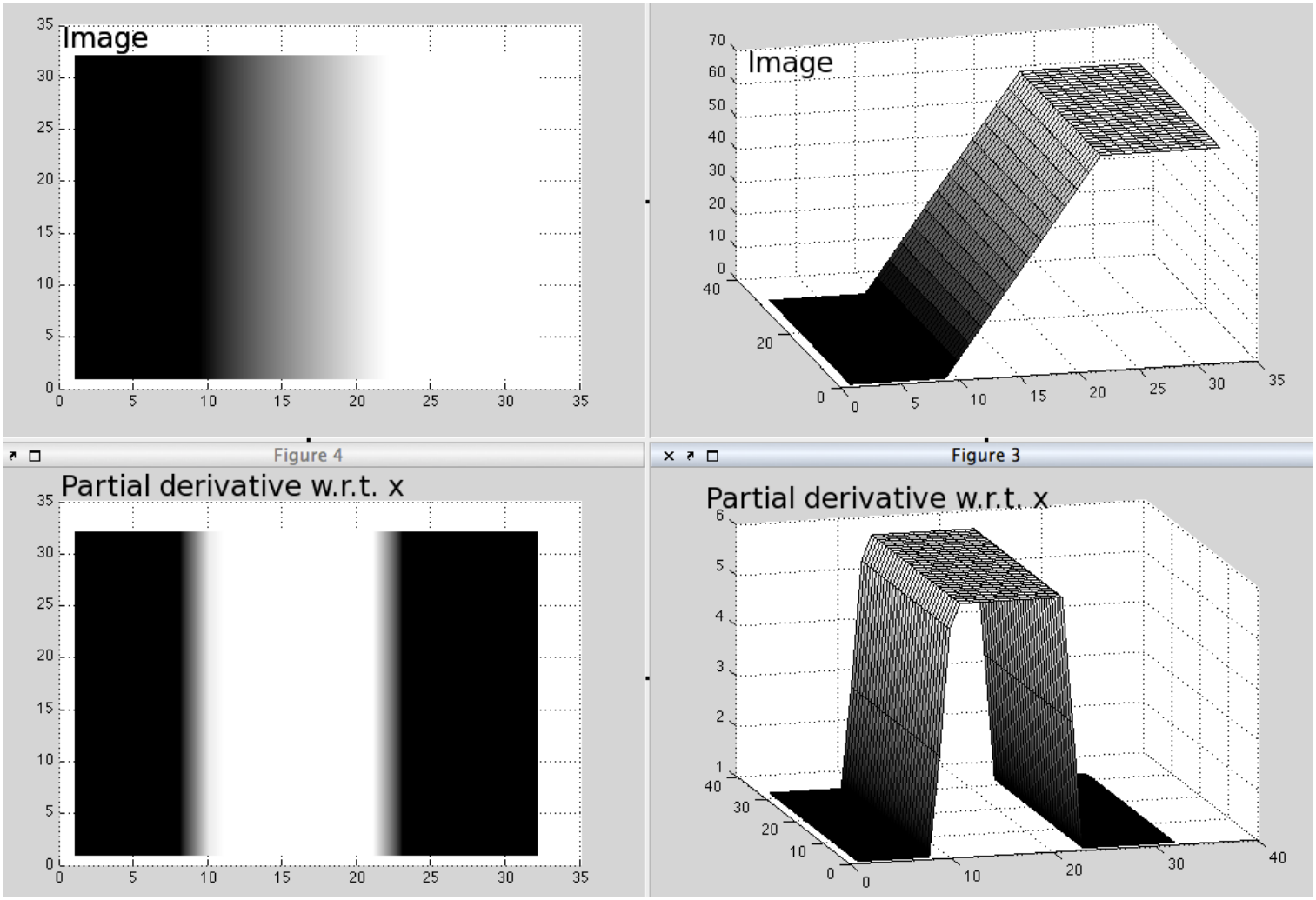
$$\int_{\Omega} |u - f|^2 dxdy$$

- $f(x,y) : \Omega \to \mathbb{R}$, a given image
- $u(x,y) : \Omega \to \mathbb{R}$, a given Image

What is it measured by this functional? It gives a number indicating a sort of distance or difference between two images. We have information about how much they differ but we do not know where.

What can we do with functions?

Example of Derivative



Could someone tell me how the Partial Derivative w.r.t. *y* is?

What can we do with functions?

Functional

- We can build a function of functions, which frequently called a functional.
- In our course, a functional will be a correspondence which assigns a finite (real) number to each **function** belonging to some class or space of functions.
- Thus, one might say that a functional is a kind of function, where the independent variable is itself a function.

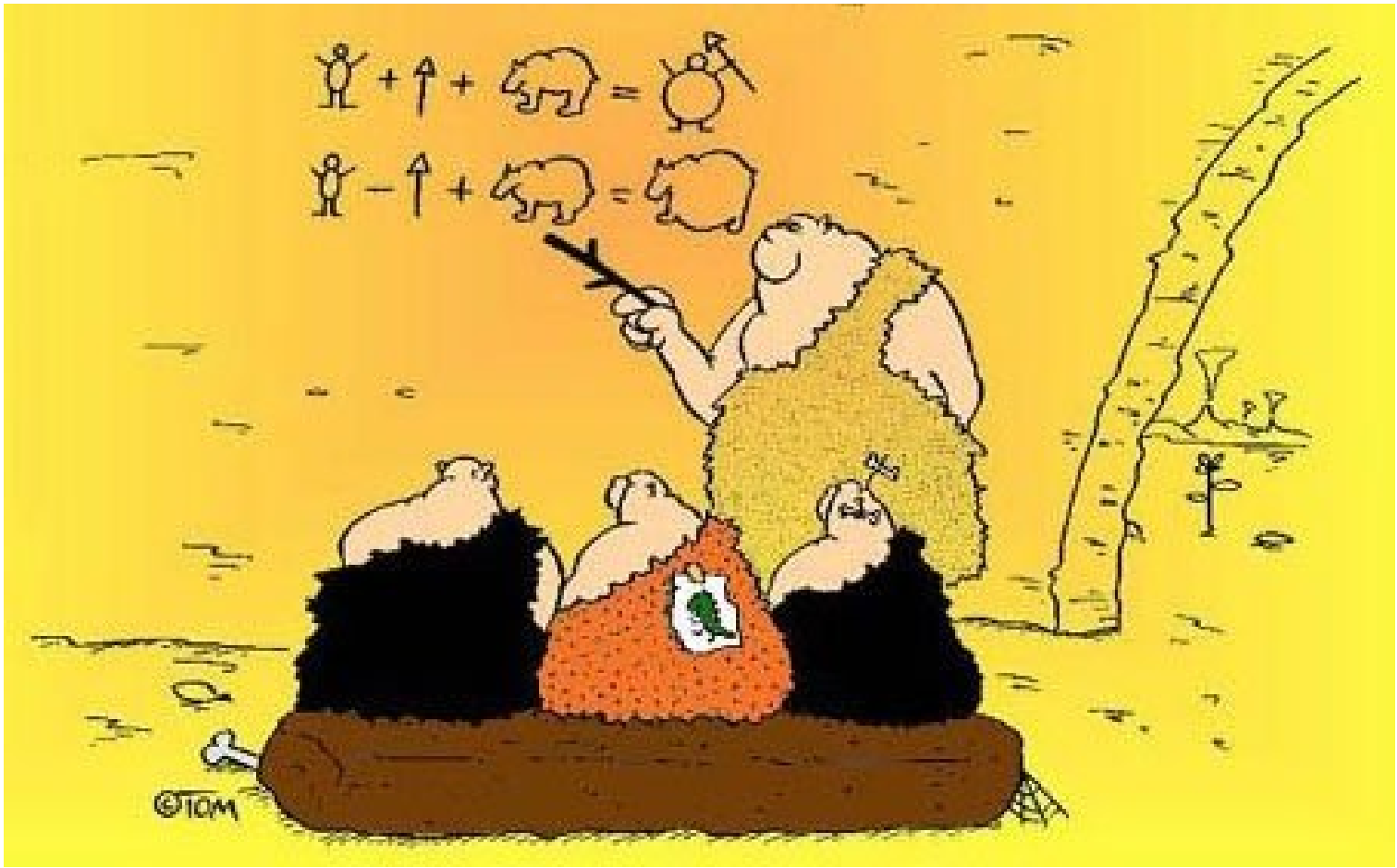
Example 1

$$J(u) = \int_{\Omega} u(x, y) dxdy$$

Example 2

$$J_f(u) = \int_{\Omega} |u(x, y) - f(x, y)|^2 dxdy$$

Modelling Real World Problems as Minimization Problems



Optimization: A General Variational Approach

Posing a (real world problem or task as a function estimation problem through a variational approach

Goal: find a function $\mathbf{u}(\mathbf{x})$ satisfying **suitable constraints**.
The constraints are formulated through an **energy** $\mathbf{J}(\mathbf{u})$ (also called a **cost/objective/gain**).

The solution of the problem/goal is the function $\mathbf{u_0}(\mathbf{x})$ that minimizes the energy $\mathbf{J}(\mathbf{u})$ (or minimizes the cost, or maximizes the gain).

In general $\mathbf{u_0}(\mathbf{x})$ optimizes $\mathbf{J}(\mathbf{u})$.

$$\mathbf{u_0} = \arg\min_{\mathbf{u}} \mathbf{J}(\mathbf{u})$$

Example: Modelling Image Restoration

Image Restoration: Image Restoration is the task of, given a corrupt/**noisy** image *f*, to estimate a clean image *u* that is similar to *f* and "without" noise.

Example: Modelling Image Restoration

First, let's deduce a modelling considering a discrete image defined on discrete pixels.

How do we measure similarity?

We want u to be *similar* to f,
$\Rightarrow$ *their values should not vary too much*.

Let's focus on a particular pixel (i, j), let $f_{i,j}$, $u_{i,j}$ be the values for the **noisy** and the **denoised images** respectively.

We should have $f_{i,j} \approx u_{i,j}$... ($f_{i,j} = u_{i,j} +$ noise)

Or in other words, $f_{i,j} - u_{i,j}$ should be *small* (although not 0!).

How do we control that?

- Absolute value: $|f_{i,j} - u_{i,j}|$ (good but not differentiable).
- Absolute value: $(f_{i,j} - u_{i,j})^2$ (good and differentiable).

We want this for all the pixels so let's just sum over all of them:

$$J(u) = \sum_i \sum_j (f_{i,j} - u_{i,j})^2$$

Is this enough?

We said we will obtain our solution optimizing the function $J(u)$ (i.e. finding its minimum).

What is

$$\arg\min_u J(u) = \arg\min_u \sum_i \sum_j (f_{i,j} - u_{i,j})^2 \ ?$$

It would be just $u = f$! *OK, we need to work a bit more...*

But wait, what if we fix a value $\epsilon > 0$ and we stop our optimization when $J(u') < \epsilon$?
Then our new solution u' would not be equal to f.

Well, that would be actually clever...
But we would not have any guarantee on if our solution would still be "an image".

How do we guarantee that we have a denoised image?

Images in absence of noise **are mostly composed by flat areas from different objects with different intensity values**.

This means that in most parts of the image local intensity varies very little and we only have big variations in the interfaces of the objects (the edges!)

How do we know about the variation or the change in functions?

We look at the gradient! (images are 2D functions (object))

$\Rightarrow$ If an image is mostly flat in an area, its gradient there will be close to 0.
$\Rightarrow$ If there is an edge in the image, its gradient there will be *big*.
$\Rightarrow$ **If there is noise in the image, its gradient in flat areas will be *small but not 0***.

How do we guarantee that we have a denoised image?

What is the gradient of an image?

We will work with a discretization of the gradient which is nothing more than the (forward) differences with the adjacent pixels.

We define the discrete gradient of a 2D image u denoted by ∇u (sometimes by $\nabla^h u$)

$$\nabla u = \begin{pmatrix} \partial_i u \\ \partial_j u \end{pmatrix} = \begin{pmatrix} (u_{i+1,j} - u_{i,j})/h \\ (u_{i,j+1} - u_{i,j})/h \end{pmatrix}$$

(where h can be taken as 1 in this case)

How do we guarantee that we have a denoised image?

OK, we have the gradient of the image...

How can we guarantee our solution is denoised?

If u does not have noise and f has, ∇u will be in some sense smaller than ∇f.

We can measure how big is the gradient of a whole image by summing over all the pixels the square of the gradient value:

$$J_S(u) = \sum_i \sum_j (\partial_i u)^2_{i,j} + (\partial_j u)^2_{i,j}$$

Image denoising algorithm

Proposed solution: Denoise f by finding a solution u that minimizes

$$J(u) = \sum_i \sum_j (\partial_i u)^2_{i,j} + (\partial_j u)^2_{i,j} + \lambda \sum_i \sum_j (f_{i,j} - u_{i,j})^2$$

where λ is a given parameter that controls the trade-off between:

- **Data fidelity term** ($J_D(u) = \sum_i \sum_j (f_{i,j} - u_{i,j})^2$), which measures the **similarity between the two images** (it gives a number measuring how much u and f differ but we do not know where).
- **Smoothness or regularization term** ($J_S(u) = (\partial_i u)^2_{i,j} + (\partial_j u)^2_{i,j}$). Minimizing this term avoids the irregularities or pixel changes due to high frequency noise.

Written in non-discrete way: Image restoration problem

Given f, the (damaged) image to restore, we describe the characteristics of the (unknown) restored image u, the solution.

- Similar to the original image f: Overall difference between u and f is small.
- Without noise: The noise changes very quickly from one pixel to the next one. So the solution should be "smooth".

Functional

$$J(u) = \int_\Omega |\nabla u|^2 dx + \lambda \int_\Omega |u - f|^2 dx$$

$$J(u) = \underbrace{\int_\Omega |\nabla u|^2 dx}_{\text{Regularity of solution}} + \underbrace{\lambda \int_\Omega |u - f|^2 dxdy}_{\text{Fidelity to data}}$$

- $f(x) : \Omega \to \mathbb{R}$, **KNOWN**. The **given** noisy image

This is a general strategy!

Posing a task as a function estimation problem through a variational optimization approach from a given f.

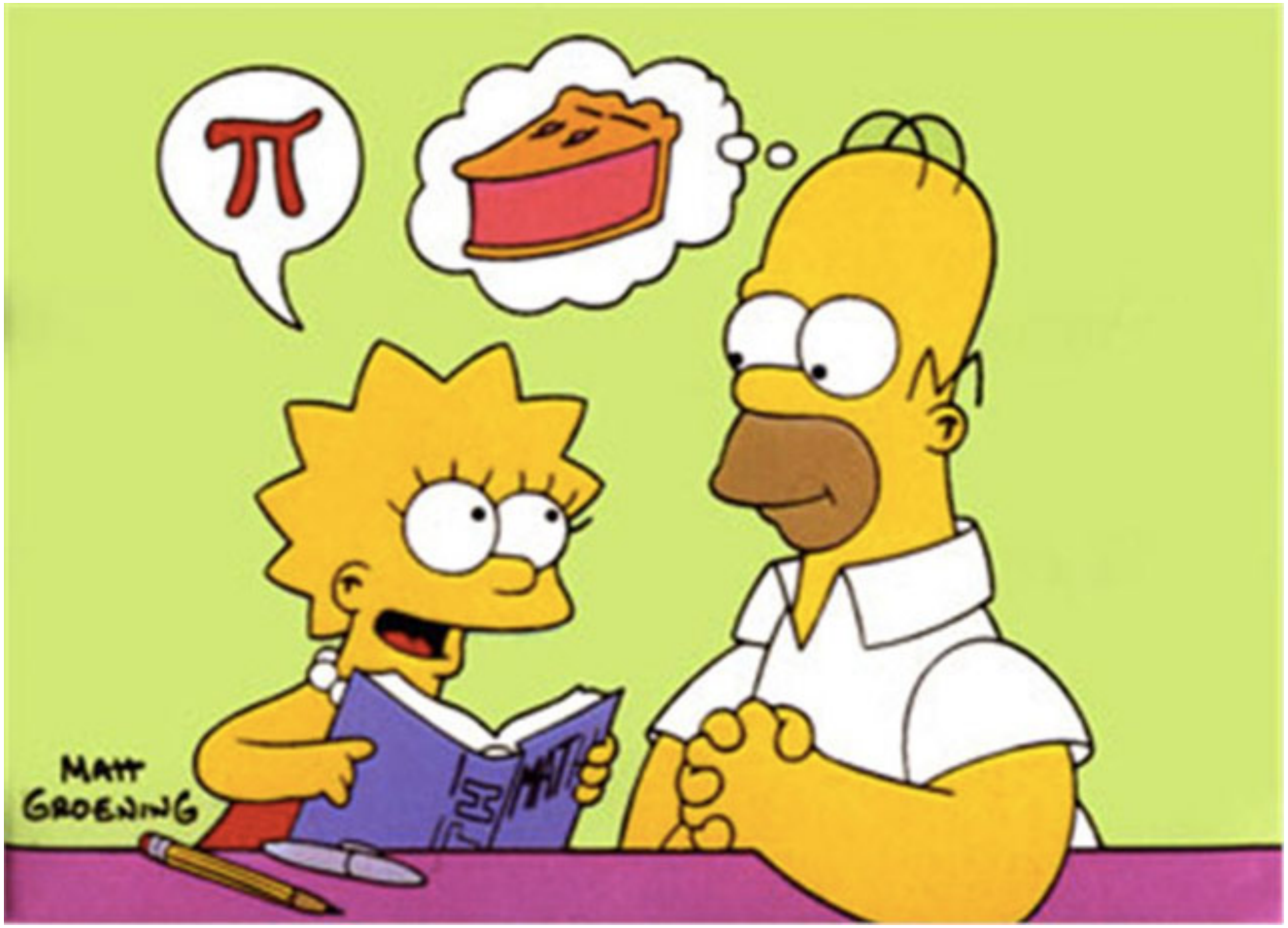
We will obtain a solution u such that:

- **Data fidelity term:** u is similar to f, *overall difference between u and f is small* ($\mathbf{J_D(u, f)}$).
- **Smoothness or regularization term:** u has certain properties we know a priori ($\mathbf{J_S(u)}$).

Proposed solution:
We can compute u by **minimizing** the functional

$$\mathbf{J(u) = J_D(u, f) + J_S(u)}$$

Formally



Modelling

The fundamental problem of calculus of variations

Let

$$\begin{aligned} J: \quad V &\to \mathbb{R}, \\ u &\mapsto J(u) = \int_\Omega \mathcal{F}(x, u(x), \nabla u(x))\, dx \end{aligned}$$

be a functional over functions u, where

- V is a suitable space of functions
- $\Omega \in \mathbb{R}^D$ is a bounded open domain on the D dimensional space $\mathbb{R}^D$.
- $u \in V : \Omega \to \mathbb{R}^N$ is a function defined on a Ω.
- ∇ is the gradient operator and $x \in \Omega$ is the spatial variable.

The fundamental problem of the calculus of variations is to find the extremum (maximum or minimum) of the functional $J(u)$.

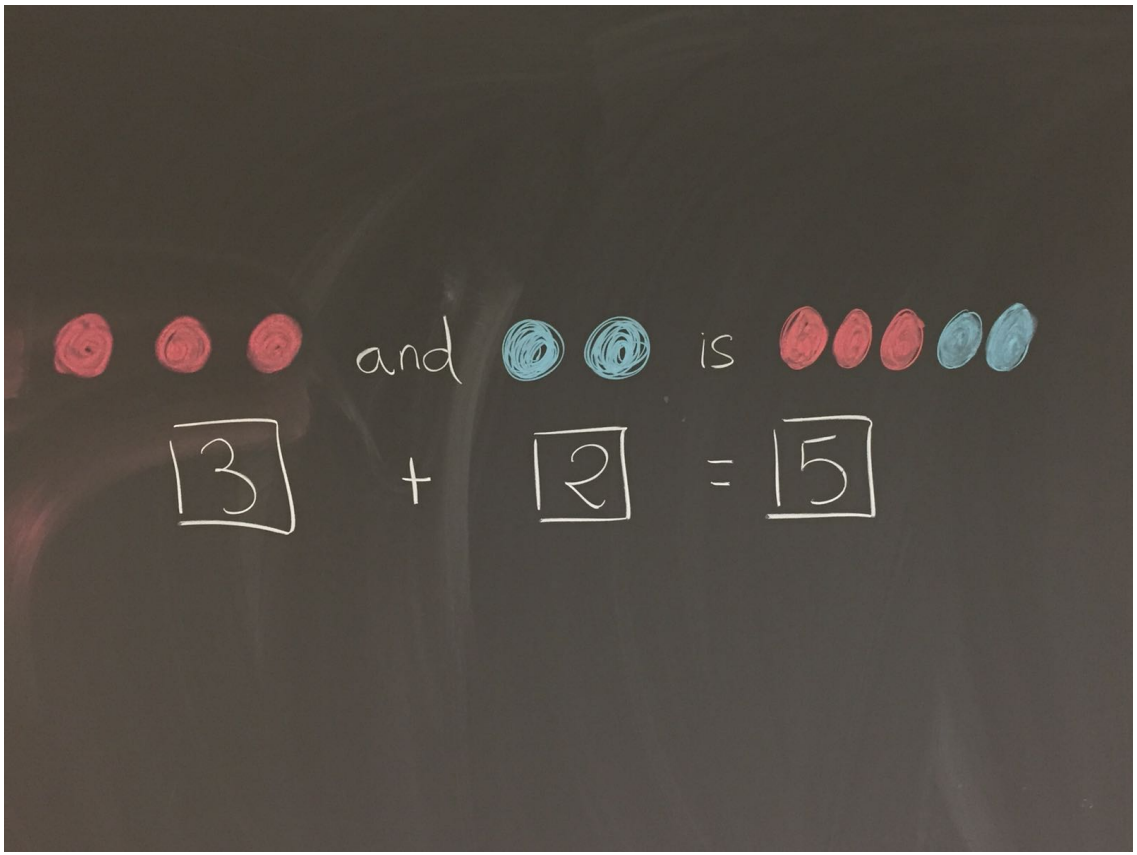
$$\arg\min_{u \in v} J(u)$$

Advantages

Modelling as Minization

Advantage

Once the problem in the real world (denoising, restoration, segmentation, optical flow.....) is modelled as a minimization problem, we can apply all the well established mathematical theory and tools to solve the problem.



Examples

(wroten in a rather formal way)

Example 1: Image Restoration

Given the image $f \in L^\infty(\Omega)$ solve

$$u = \underset{u \in W^{1,2}(\Omega)}{\arg\min} \left\{ \int_\Omega |\nabla u|^2 dxdy + \frac{1}{2\lambda} \int_\Omega |u - f|^2 dxdy \right\}$$

Where

- $W^{1,2}(\Omega) = \{u \in L^2(\Omega);\ \nabla u \in L^2(\Omega)^2\}$ is the space for u is well-defined.
  - $u \in L^2(\Omega)$ means $\int_\Omega u^2 dx < \infty$.
  - $\nabla u \in L^2(\Omega)^2$ means all derivatives up to order 1 belong to $L^2(\Omega)$
- λ Given parameter that controls the trade-off between data fidelity term and smoothness term.

Image Restoration

Given noisy image f

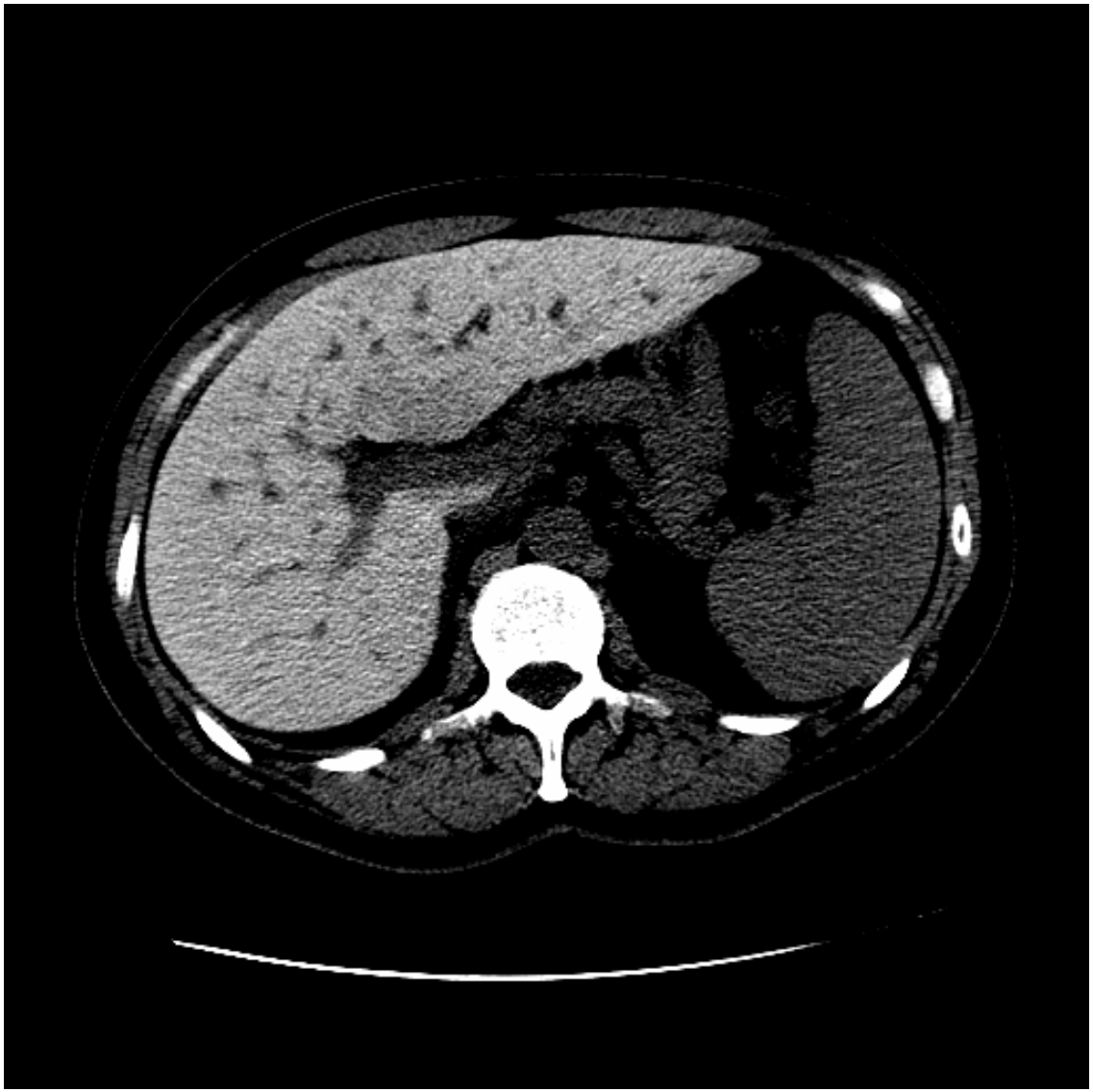


Image Restoration

Restored image u (minimum of $J(u)$)

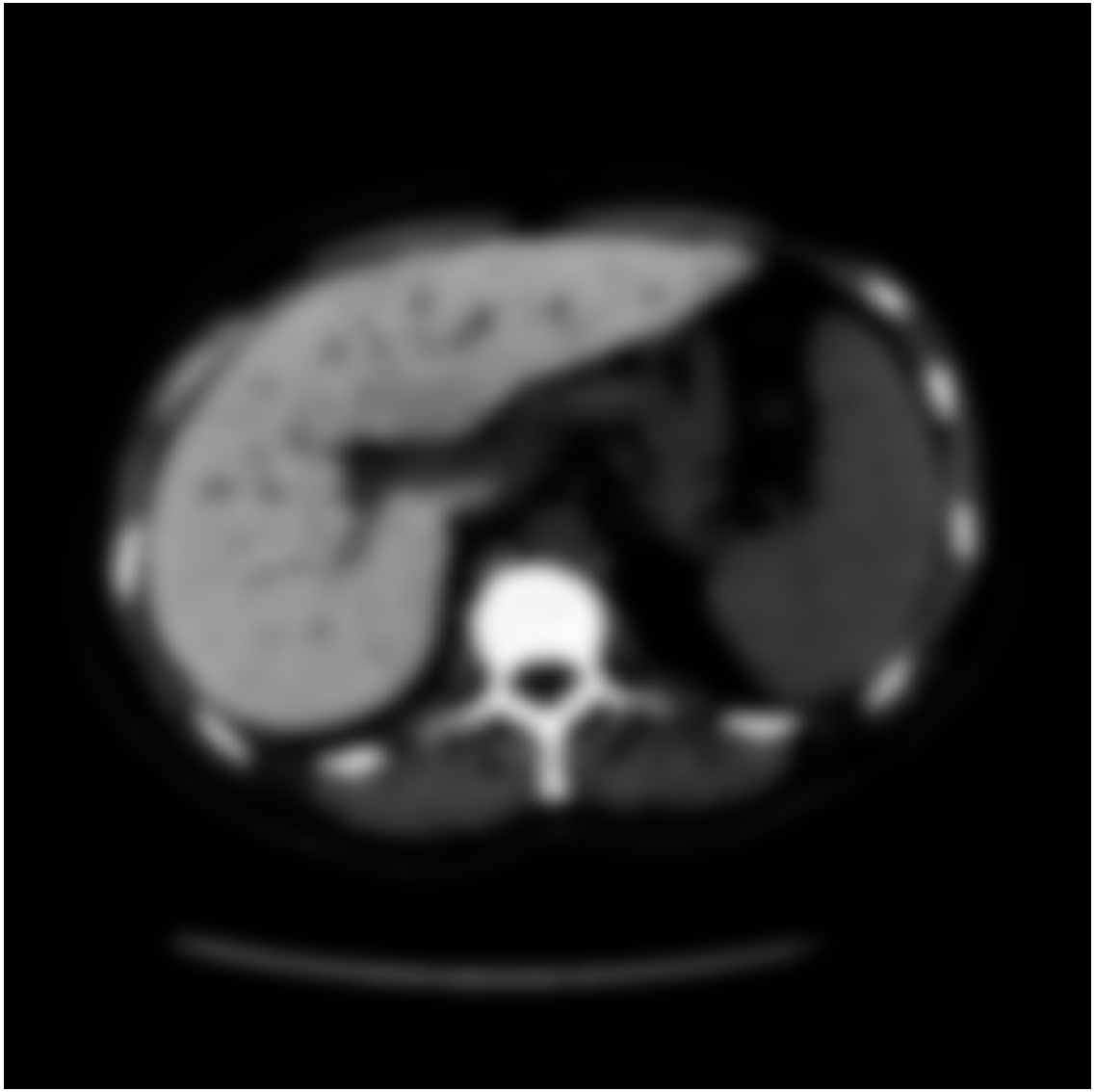


Image Restoration

Is *u* the solution to our mathematical problem (finding the minimum of the proposed energy functional)?

YES!!

but Is the solution *u* a good candidate for our restoration problem (the real world problem)?

NO!!

The boundaries of the objects have been lost.

Question

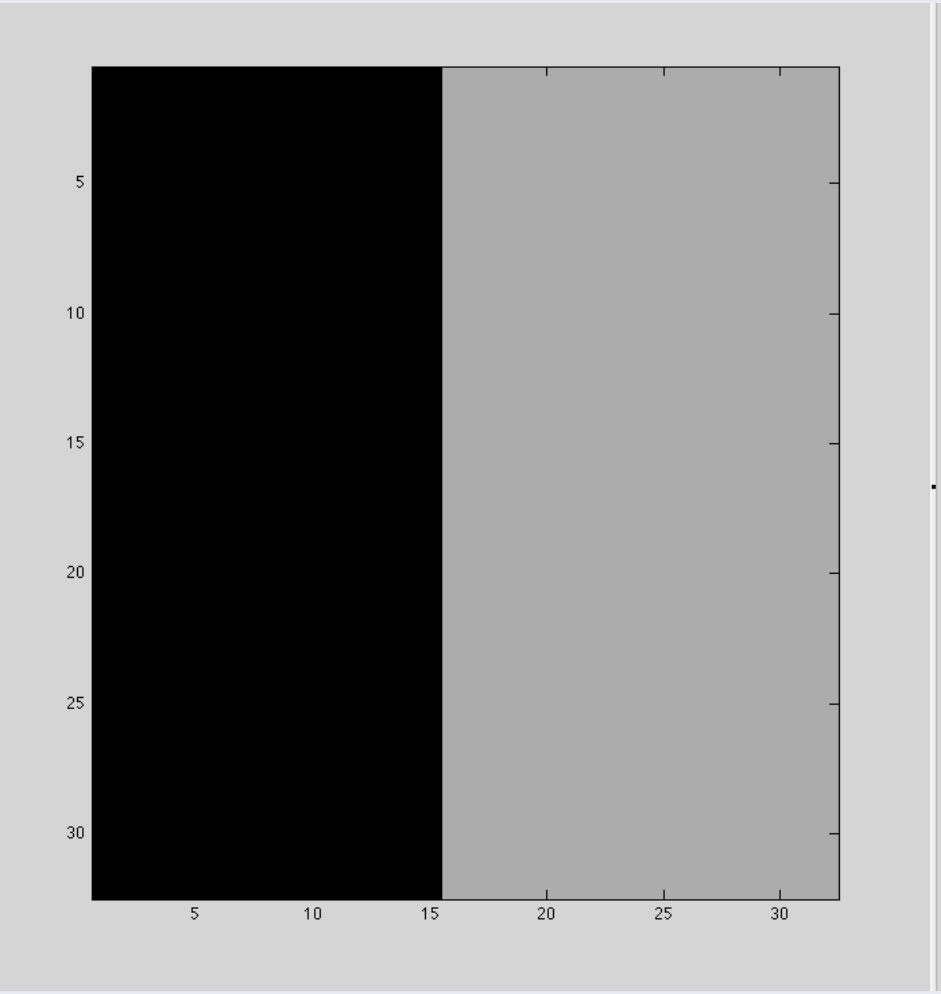
Which characteristic about regularity has the $W^{1,2}(\Omega)$ functional space?

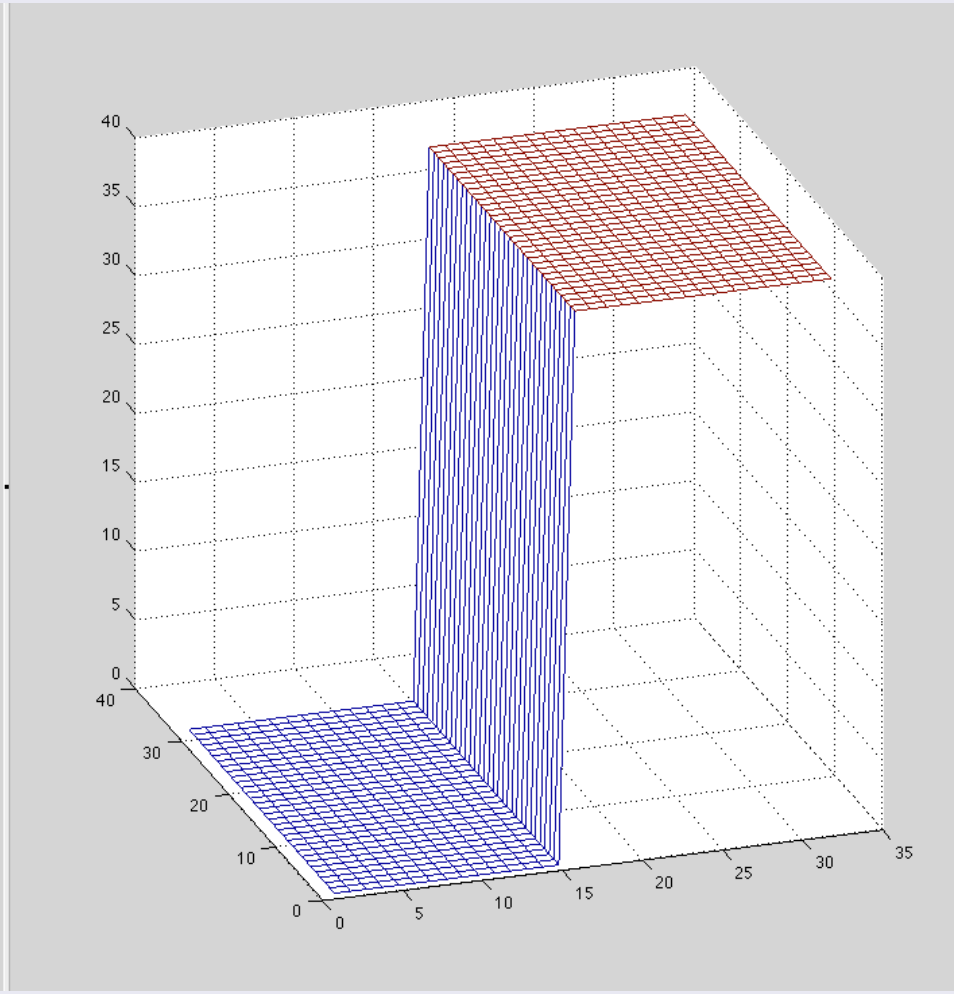
Answer

Functions with discontinuities does not belong to $W^{1,2}(\Omega)$ functional space, so it is too regular to model natural images.

Image Restoration

Could be this image a solution?





Another (and more intuitive) explanation: The L^p norm $|\cdot|^p$, with $p = 2$ of the gradient, $\int_\Omega |\nabla u|^2 dx$, penalizes too much the gradients corresponding to edges.
One should then decrease p in order to preserve the edges as much as possible.

Image Restoration

Total Variation, $p = 1$

$$\int_{\Omega} |\nabla u|^p dxdy$$

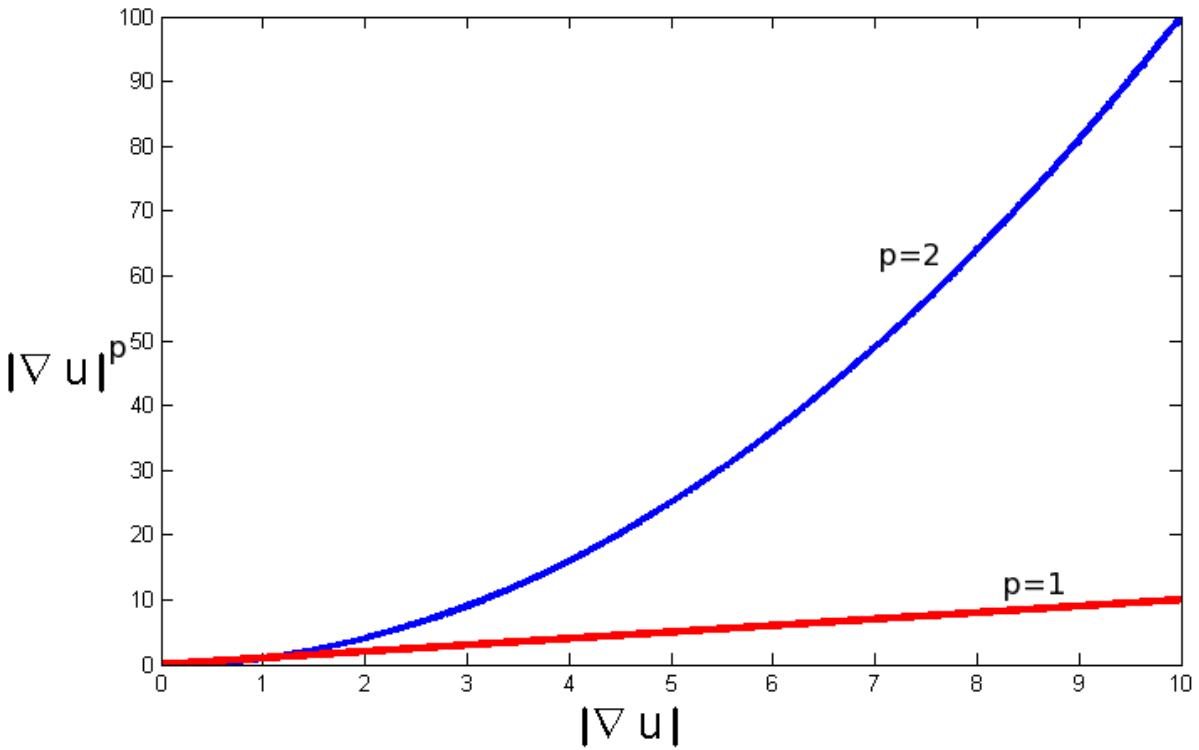
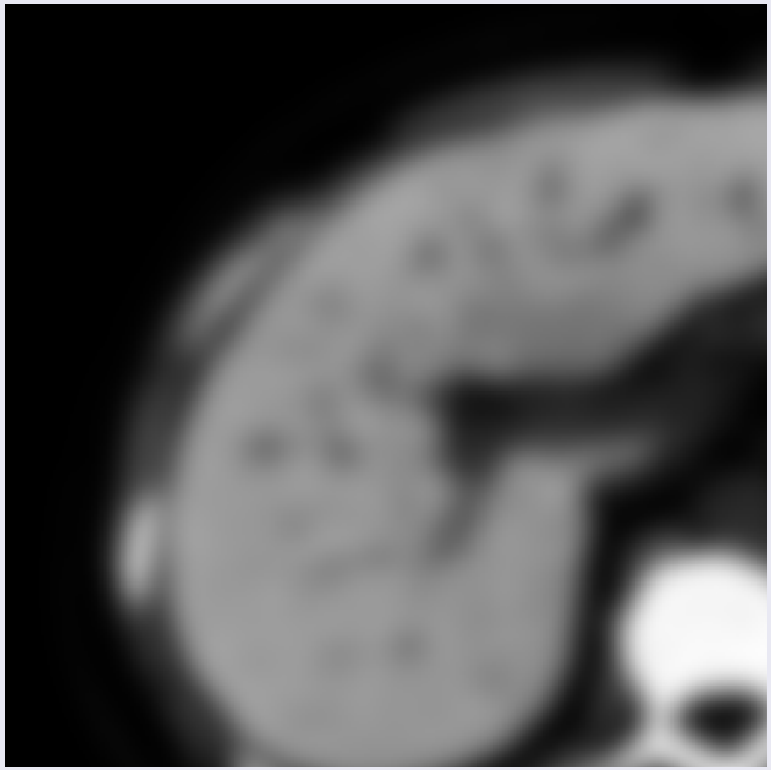
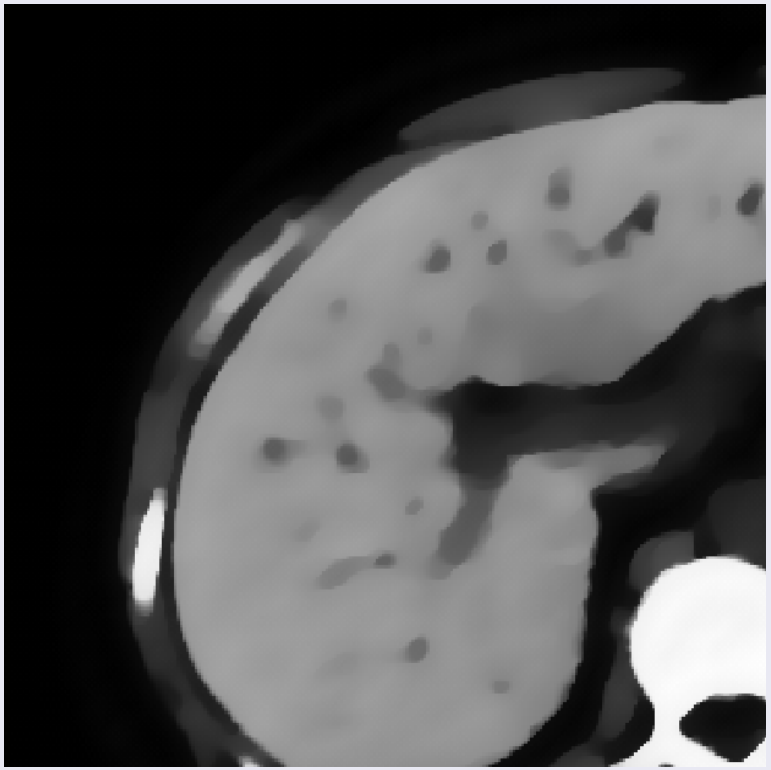


Image Restoration

p=2



p=1



Take home message

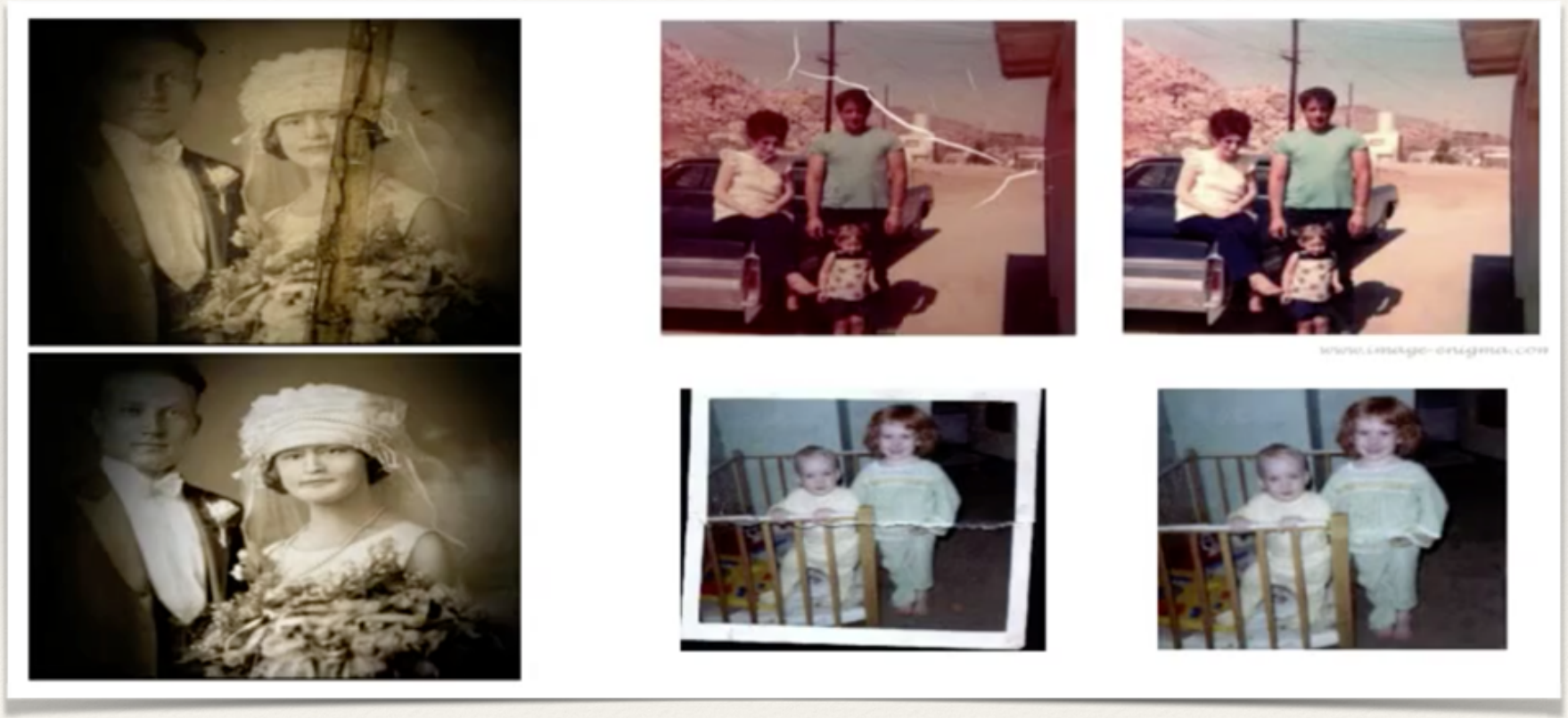
- A problem can be modelled in many different ways.
- The solution of the model may be is not the solution to your problem.

Example 2: Image inpainting

What is Inpainting?

Detail of "Cornelia, Mother of the Gracchi" by J. Suvee (Louvre). Courtesy of Emile-Male "The Restorer's Handbook of easel painting".

What is Inpainting?



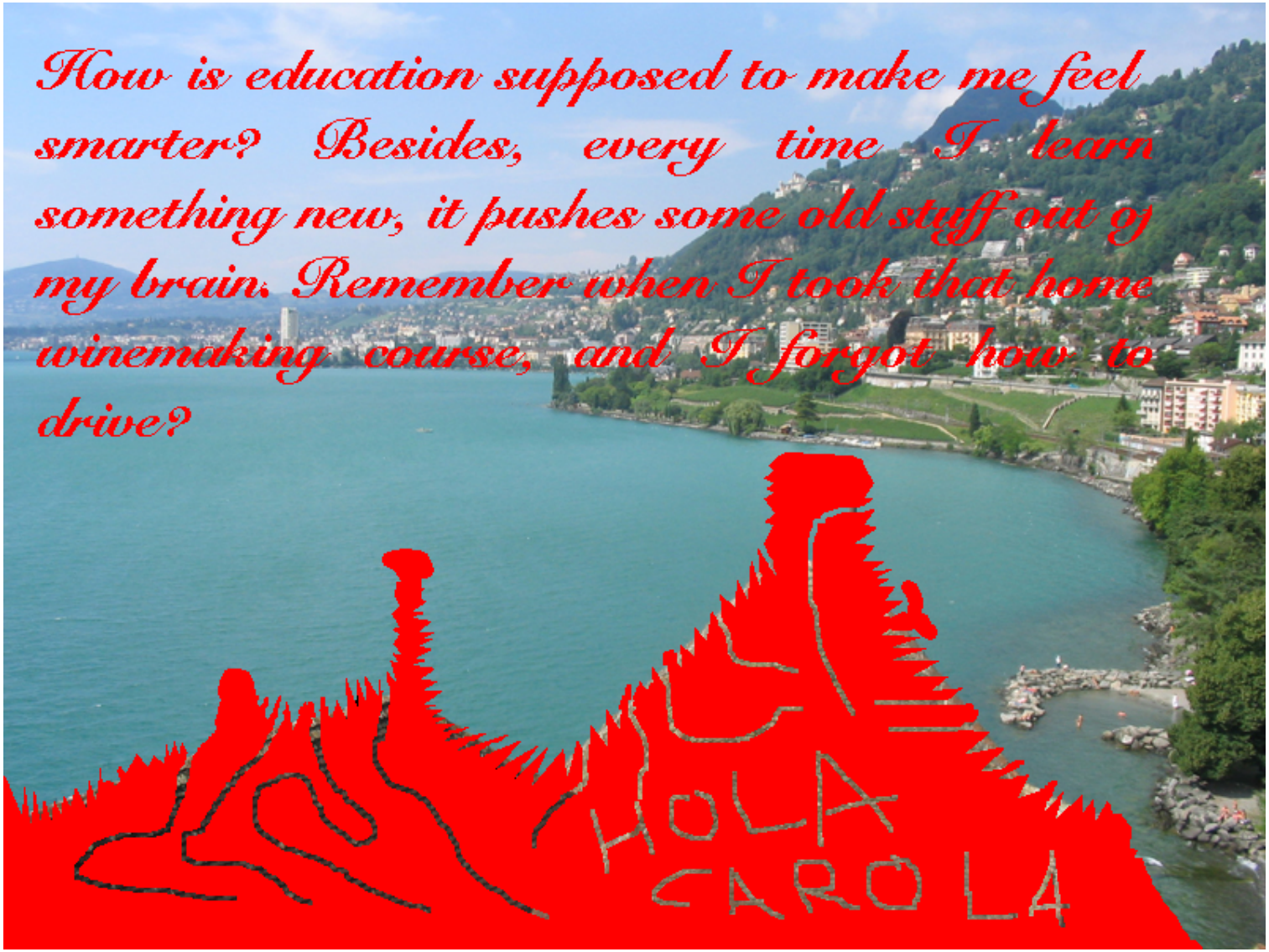
What is Inpainting?

What is Inpainting?

What is not Inpainting?

We can only obtain one of the plausible completions

Your Project

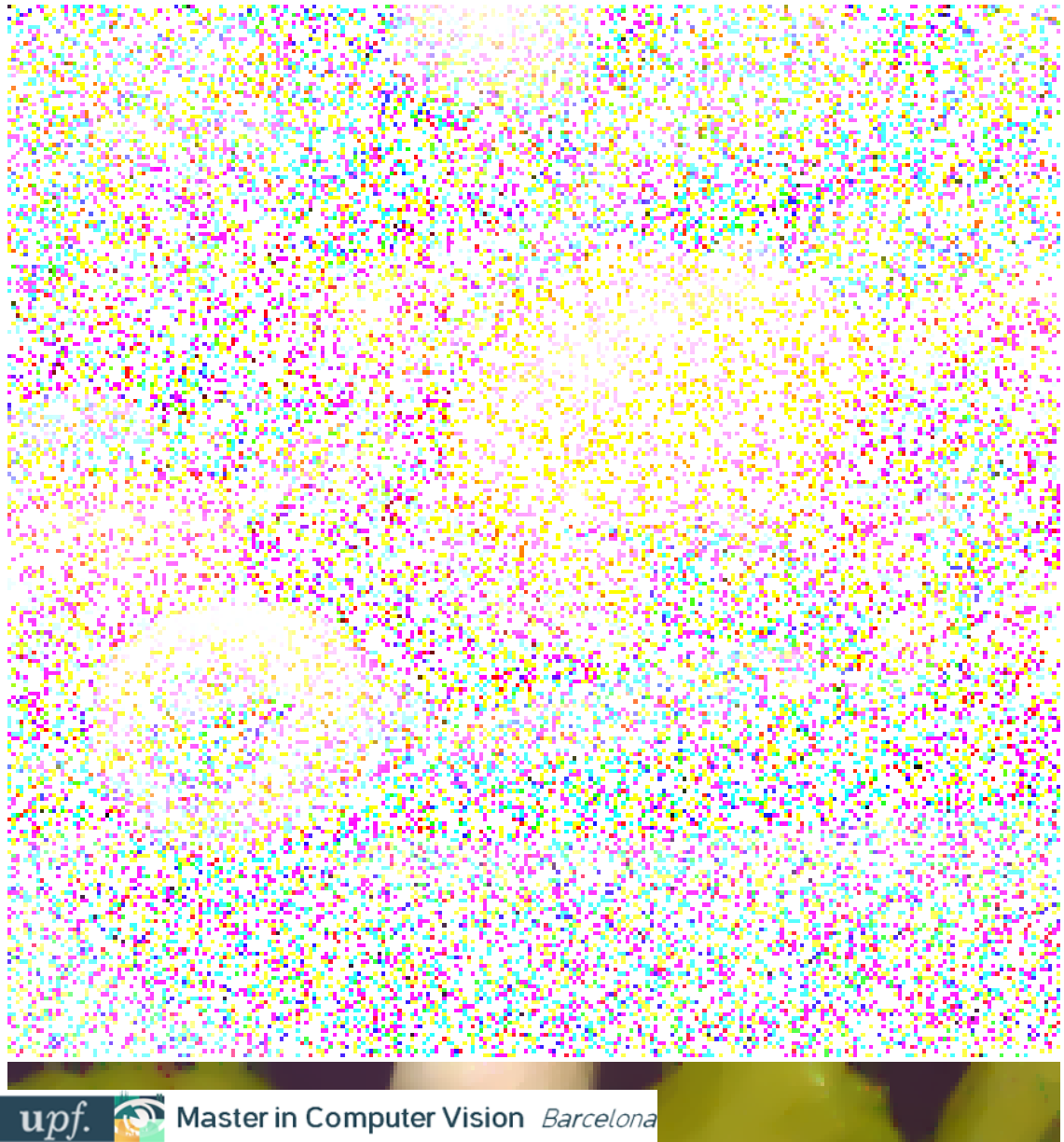


Inpainting using a first simple model

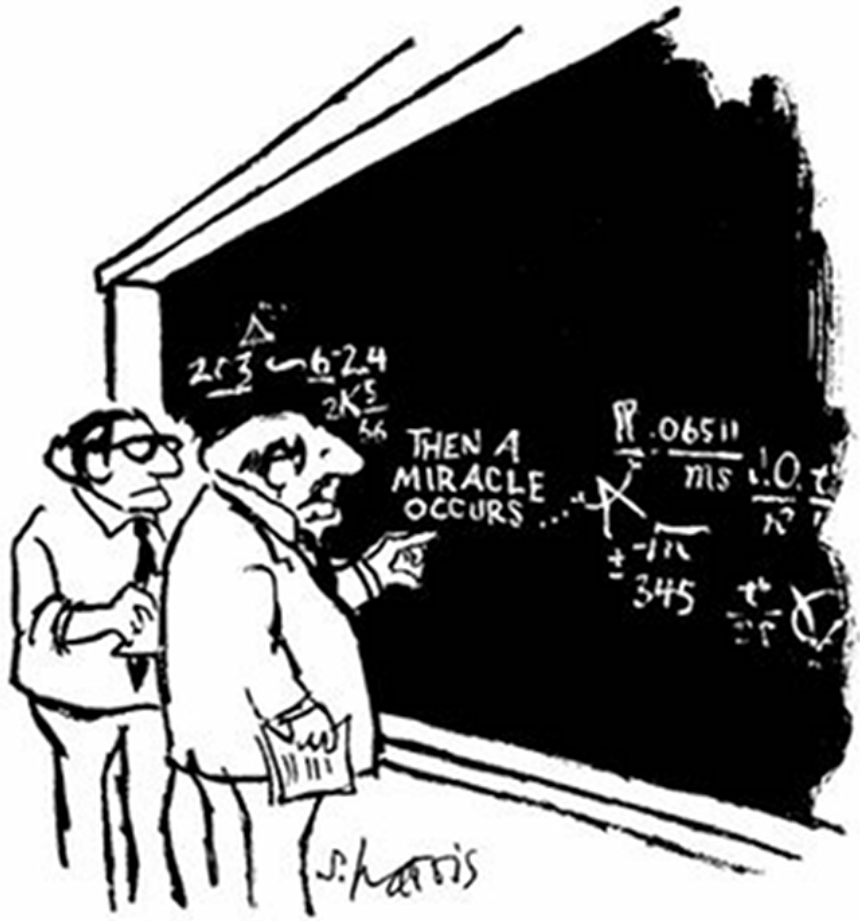
Let $f : \Omega \to \mathbb{R}$ be a given grayscale image and let $D \subset \Omega$ be an open set representing the region to be inpainted.
It is supposed that f is known in $\Omega \backslash D := \{x \in \Omega : x \notin D\}$
The inpainting solution u can be found as the minimum of

$$\begin{cases} \arg\min_{u \in W^{1,2}(\Omega)} \int_D |\nabla u(x)|^2 dx, \\ u|_{\partial D} = f \end{cases}$$

Example 3: Inpainting when the information is given on isolated points



Understanding the Miracle: From the minimization problem to image solution



"I think you should be more explicit here in step two."

The Mathematical Problem to Solve

For a given energy functional

$$J(u) = \int_{\Omega} \mathcal{F}\left(x, u(x), \nabla u(x)\right) dx$$

The fundamental problem of the calculus of variations is to find the extremum (maximum or minimum) of the functional $J(u)$.

How do we solve these problems?

How do we minimize an energy?

Once we have been able to model our problem as an energy minimization problem:

$$\mathbf{u_0} = \arg\min_{\mathbf{u}} \mathbf{J}(\mathbf{u})$$

How to minimize $J(u)$?

This course includes tools to achieve it.

We will see:
Neccesary condition for the extremum (maximum or minimum) of the functional $J(u)$. Under some assumptions, if u is an extremum, u has to satisfy an equation.

Analogy with:

$$\mathbf{x_0} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}(\mathbf{x})$$

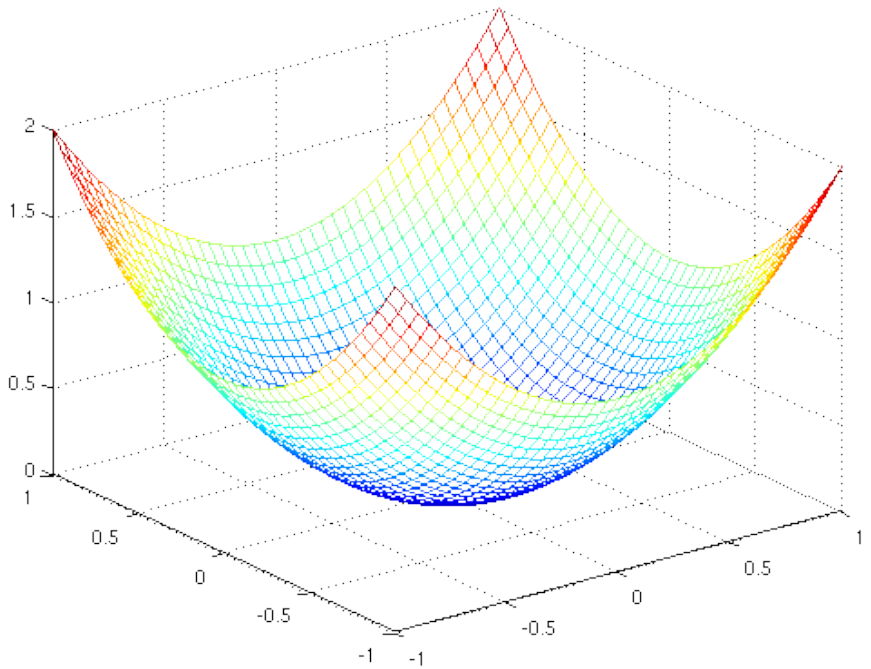
with $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}$ a smooth function.

How do we minimize an energy?

Analogy with:

$$\mathbf{x_0} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}(\mathbf{x})$$

with $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}$ a smooth function.



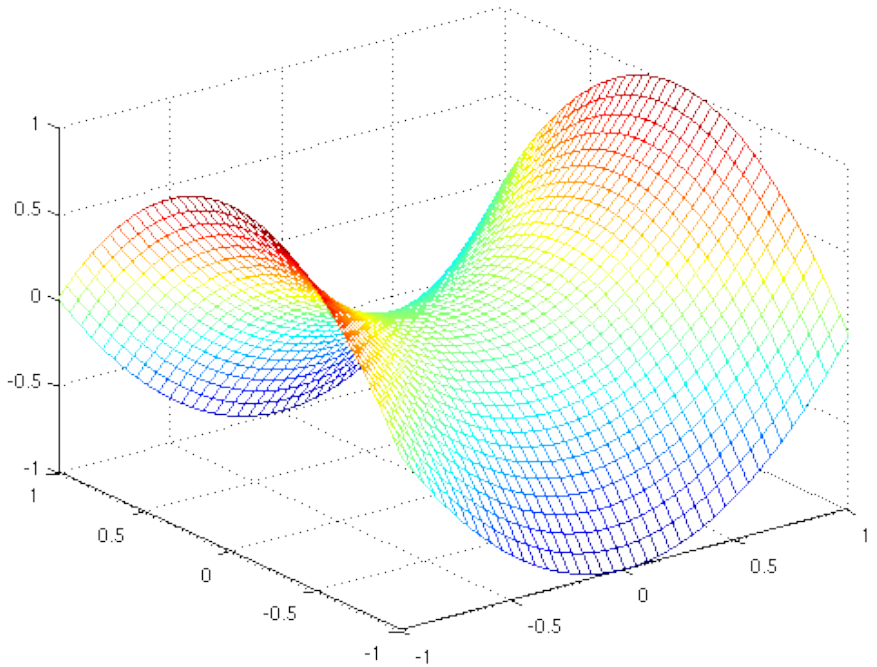
We know that the solution x_0 must have $\nabla f(x_0) = 0$.

How do we minimize an energy?

Analogy with:

$$\mathbf{x_0} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{f}(\mathbf{x})$$

with $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}$ a smooth function.



We know that the solution x_0 must have $\nabla f(x_0) = 0$ but ...

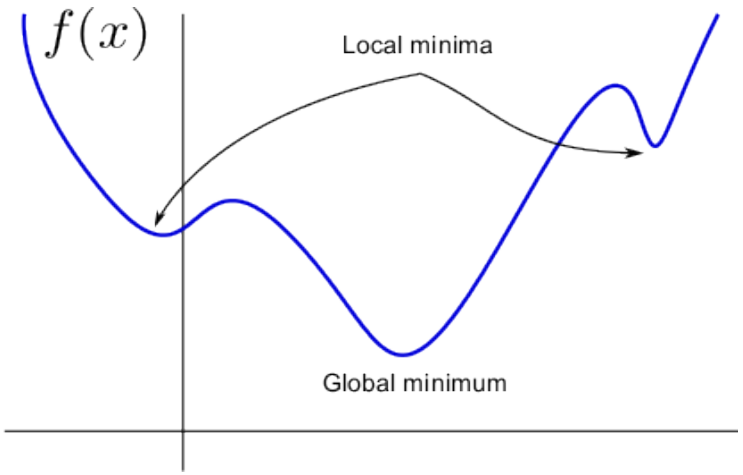
Global vs Local Minimization

We would like to find a **global minimum** of f, that is, a point $x^* \in R^n$ such that

$$f(x^*) \leq f(x) \quad \forall x \in R^n$$

Optimization algorithms usually only explore a finite number of points. Most algorithms can only find a **local minimum** of f, that is, a point x^* such that

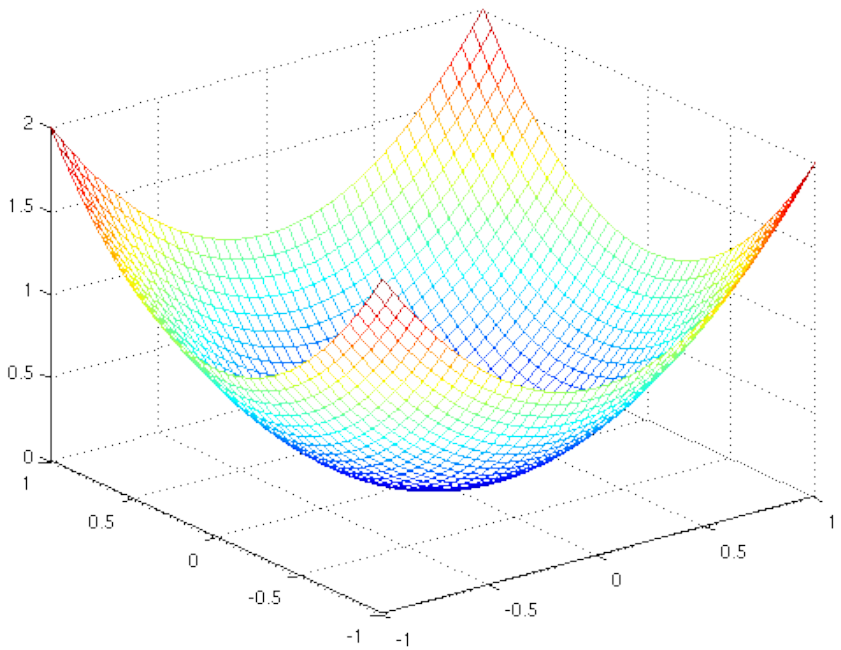
$$f(x^*) \leq f(x) \quad \forall x \in \text{Neighborhood}(x^*)$$



Our favorite: Convex Optimization

If f is **convex**, there are minima!

If f is **convex** (and some more hypothesis that we will see):
$\Rightarrow$ There exists a global minimum of f!
$\Rightarrow$ A point $\mathbf{x_0} \in \mathbb{R}^n$ such that $\mathbf{x_0} = \arg\min_{x \in \mathbb{R}^n} f(x)$.
Moreover, the global minimum $\mathbf{x_0}$ is unique if f is strictly convex!
(nice, because $\mathbf{x_0}$ will be the global minimum).



Review on Differential calculus

First order necessary conditions.

For a given function $f(x) : \mathbb{R}^n \to \mathbb{R}$, if x_0 is a local minimum or maximum (extremum) then $\nabla f(x_0) = 0$.

Second order sufficient conditions.

For a given function $f(x) : \mathbb{R}^n \to \mathbb{R}$, if $\nabla f(x_0) = 0$ and $D^2 f(x_0))$ is positive definite then x_0 is a local minimum.

(If $\nabla f(x_0) = 0$ and $D^2 f(x_0)$ negative definite, then x_0 is a local maximum.).

Main intuitions

For a given energy function, in order to obtain a (candidate of) solution of the associated minimization problem

$$\min_x f(x),$$

two main strategies are used:

- Solving the necessary condition of extremum

$$\nabla f(x_0) = 0$$

- Using an iterative algorithm converging to the extremum: Line search methods. The gradient descent is probably the most used.

Parenthesis: Line search methods

The *line search methods* calculate *stationary points* from differentiable functions $f : \mathbb{R}^n \to \mathbb{R}$. They are iterative methods that start from an *initial point* $\mathbf{x}_0 \in \mathbb{R}^n$, and generate a succession of *iterates*:

$$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k, \mathbf{x}_{k+1}, \ldots \quad \in \mathbb{R}^n.$$

The goal is to find a succession of points converging to a stationary point of the objective function f:

$$\lim_{k \to \infty} \mathbf{x}_k = \mathbf{x}^* \quad \text{such that} \quad \nabla f(\mathbf{x}^*) = \mathbf{0}.$$

The iterate $\mathbf{x}_{k+1}$ is calculated from $\mathbf{x}_k$ and local information from the function f in a neighborhood of $\mathbf{x}_k$. The line search methods calculate $\mathbf{x}_{k+1}$ as:

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

where

- $\mathbf{p}_k \in \mathbb{R}^n$ is the *search direction*.
- $\alpha_k > 0$ is the *step length*.

Search direction and Wolfe conditions

There are several *line search methods*, depending on how are $\mathbf{p}_k$ and α_k calculated. We will focus on two of the most popular methods, the *gradient descent* and the *Newton's method* (and some of their variants). Let us first consider a general problem for all these methods:

Which conditions can we impose on the directions $\mathbf{p}_k$ and the step lengths α_k to guarantee the convergence of the method to a stationary point?

Descent directions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function ($\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is continuous).

We say that $\mathbf{p} \in \mathbb{R}^n$ is a *descent direction* of f in $\mathbf{x}$ if

$$\langle \mathbf{p}, \nabla f(\mathbf{x}) \rangle < 0.$$

Then, if $\mathbf{p}$ is a *descent direction*,

$$f(\mathbf{x}) > f(\mathbf{x} + \alpha \mathbf{p})$$

for a small enough value of α. This is, we can find lower values of f if we move in the direction of $\mathbf{p}$.

This is a direct application of a first order Taylor approximation:

$$f(\mathbf{x} + \alpha \mathbf{p}) = f(\mathbf{x}) + \alpha \langle \mathbf{p}, \nabla f(\mathbf{x}) \rangle + o(\alpha),$$

for a small value of α, the first order term dominates the residual $o(\alpha)$.

Parenthesis: Choosing a step length

How to choose the *step length* α: We denote by $\phi(\alpha) = f(\mathbf{x} + \alpha\mathbf{p})$.

The simplest option is to use a *fixed step length*, $\alpha = \delta$ in every iteration. However, the step length has implications on the convergence and the speed of convergence. Option very easy to implement but very limited in practice.

Ideally, we would like to minimize as much as possible the objective function in the search direction $\mathbf{p}$. The *exact line search* is based on calculating the *optimal step length* α^*:

$$\alpha^* = \arg\min_{\alpha>0} \phi(\alpha) = \arg\min_{\alpha>0} f(\mathbf{x} + \alpha\mathbf{p})$$

However, calculating the optimal step length is normally too computationally expensive.
If α^* is a minimum of ϕ, then:

$$\phi'(\alpha^*) = 0 \quad \underset{\text{chain's rule}}{\Leftrightarrow} \quad \langle \nabla f(\mathbf{x} + \alpha^*\mathbf{p}), \mathbf{p} \rangle = 0.$$

The optimal step length is given in points in where the gradient of f is orthogonal to the direction of p. In other words, when the search direction is tangent to a level line.

Parenthesis: Gradient descent method

The ***gradient descent method*** *or* ***steepest descent*** is a search line method with $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$, the direction of maximum decreasing of f in $\mathbf{x}_k$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

To set the step length all the previously mentioned alternatives can be used: fixed, optimal and approximated.

The gradient descent is one of the simplest and most used minimization methods but it is also one of the slowest.

Coming back to our "functional" case: The Mathematical Problem to Solve

For a given energy functional

$$J(u) = \int_{\Omega} \mathcal{F}\left(x, u(x), \nabla u(x)\right) dx$$

The fundamental problem of the calculus of variations is to find the extremum (maximum or minimum) of the functional $J(u)$.

Solving the Mathematical Problem

There are two main strategies.

- First strategy: Finding the necessary condition for the extremum
  - The change of the functional with respect to the unkown/s has/have to be zero. i.e, it's derivative w.r.t. u has to be zero.
  - So, under some assumptions, if u is a extremum, u has to satisfy the equation.

$$\frac{dJ(u)}{du} = 0$$

  - The second strategy uses also $\frac{dJ(u)}{du}$ but in an iterative Gradient Descent algorithm.
    The first strategy is usually more difficult to implement than the second (Gradient Descent-based) one, but usually faster.

Solving the Mathematical Problem

- Second strategy: Based on Gradient Descent algorithm:
  - Start from some initial solution u_0.

$$u^{[0]} = u_0$$

  - Iteratively go to another solution $u^{[k+1]}$, following a descent direction *dir* that is far from the previous solution $u^{[k]}$ by a distance τ and has lower energy.

$$u^{[k+1]} = u^{[k]} + \tau dir$$

  - Remember, gradient points to the direction of maximum change. Do previous step following the (some kind of generalization) gradient descent direction ($dir = -\frac{d\,J(u)}{d\,u}$ of your energy functional.

$$\begin{cases} u^{[0]} & = & u_0 \\ u^{[k+1]} & = & u^{[k]} - \tau \frac{d\,J(u)}{d\,u} \end{cases}$$

Solving the Mathematical Problem

On both strategies

We have to measure the rate of change (i.e. compute the derivative) of the energy functional $J(u)$ with respect to the unknown u.
Have you noticed that u is a function?

It’s a complex business to derive w.r.t. functions!!

Remember Definition of Derivative

1D Derivative

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

Multidimensional Derivative

$$D_{\vec{h}} f(\vec{X}) = \lim_{\epsilon \to 0} \frac{f(\vec{X} + \epsilon \vec{h}) - f(\vec{X})}{\epsilon}$$

and $D_{\vec{h}} f(\vec{X}) = < \nabla f(\vec{X}), \vec{h} >$.

For functionals

We do something similar: The Gâteaux Derivative

$$\left. \frac{d\,J}{d\,u} \right|_h = \lim_{\alpha \to 0} \frac{J(u + \alpha h) - J(u)}{\alpha}$$

The Gâteaux Derivative

The Gâteaux derivative extends the concept of directional derivative (in differential calculus) to infinite-dimensional spaces.
The derivative w.r.t. u of the functional J(u) in the direction $h(x)$ is defined as

$$\left.\frac{d\,J}{d\,u}\right|_h = \lim_{\alpha \to 0} \frac{J(u+\alpha h) - J(u)}{\alpha}$$

IMPORTANT: Now, the direction is a FUNCTION!

Example:

Denoising with the L^2 norm regularity term

Example

Denoising with the L^2 norm regularity term

The problem of denoising a given image is modelled as the minimization of

$$J(u) = \frac{1}{2}\int_\Omega |\nabla u|^2 dxdy + \frac{1}{2\lambda}\int_\Omega |u - f|^2 dxdy$$

Let's compute the Gateaux derivative on this example.

Example

Denoising with the L^2 norm regularity term

Let's compute $\frac{d\,J(u)}{d\,u}$. Let me denote it by $D_h J(u)$, by analogy with the case on a function $f : \mathbb{R}^n \to \mathbb{R}$ where we have

$$D_{\vec{h}} f(\vec{X}) = < \nabla f(\vec{X}), \vec{h} >$$

and we use the simple strategy of the relationship of the directional derivative and its relation with the gradient:

Strategy: Take any direction h and compute

$$\mathbf{D_h f(X)} = \frac{d}{d\epsilon} f(\vec{X} + \epsilon h)|_{\epsilon=0} = \lim_{\epsilon \to 0} \frac{f(\vec{X} + \epsilon h) - f(\vec{X})}{\epsilon}$$

and identify this limit as

$$D_h f(\vec{X}) = \langle \nabla f(\vec{X}), h \rangle.$$

From this, we are able to extract the gradient $\nabla f(\vec{X})$.

(On the blackboard)

Formal computations for the general Euler-Lagrange equation

The Gâteaux Derivative

Let $u \in \Omega \to \mathbb{R}$ with $\Omega = [a, b] \subset \mathbb{R}$ being an open bounded subset of $\mathbb{R}$ (i.e. u is 1D). The canonical form of a functional is

$$J(u) = \int_{\Omega} \mathcal{F}(u, u')dx$$

and the Gâteaux Derivative is given by

$$\left.\frac{dJ}{du}\right|_h = \lim_{\alpha \to 0} \frac{1}{\alpha} (J(u + \alpha h) - J(u))$$

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \left(\int_{\Omega} \mathcal{F}(u + \alpha h, (u + \alpha h)')dx - \int_{\Omega} \mathcal{F}(u, u')dx \right)$$

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \int_{\Omega} \mathcal{F}(u + \alpha h, (u + \alpha h)') - \mathcal{F}(u, u')dx$$

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \int_{\Omega} \mathcal{F}(u + \alpha h, u' + \alpha h') - \mathcal{F}(u, u')dx$$

Because of $\alpha \to 0$, we use taylor expansion of $\mathcal{F}(u + \alpha h, u' + \alpha h')$ in (u, u')

The Gâteaux Derivative

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \int_{\Omega} \mathcal{F}(u + \alpha h, u' + \alpha h') - \mathcal{F}(u, u') dx$$

Because of $\alpha \to 0$, we use taylor expansion of $\mathcal{F}(u + \alpha h, u' + \alpha h')$ in (u, u')

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \int_{\Omega} \mathcal{F}(u, u') + \frac{\partial \mathcal{F}}{\partial u} \alpha h + \frac{\partial \mathcal{F}}{\partial u'} \alpha h' + O(\alpha^2) - \mathcal{F}(u, u') dx$$

$$= \lim_{\alpha \to 0} \frac{1}{\alpha} \int_{\Omega} \cancel{\mathcal{F}(u, u')} + \frac{\partial \mathcal{F}}{\partial u} \alpha h + \frac{\partial \mathcal{F}}{\partial u'} \alpha h' + O(\alpha^2) - \cancel{\mathcal{F}(u, u')} dx$$

$$= \lim_{\alpha \to 0} \frac{1}{\cancel{\alpha}} \int_{\Omega} \cancel{\mathcal{F}(u, u')} + \frac{\partial \mathcal{F}}{\partial u} \cancel{\alpha} h + \frac{\partial \mathcal{F}}{\partial u'} \cancel{\alpha} h' + O(\alpha^{\cancel{2}\,1}) - \cancel{\mathcal{F}(u, u')} dx$$

$$= \lim_{\alpha \to 0} \frac{1}{\cancel{\alpha}} \int_{\Omega} \cancel{\mathcal{F}(u, u')} + \frac{\partial \mathcal{F}}{\partial u} \cancel{\alpha} h + \frac{\partial \mathcal{F}}{\partial u'} \cancel{\alpha} h' + \cancel{O(\alpha^{2\,1})} \to 0 - \cancel{\mathcal{F}(u, u')} dx$$

$$= \int_{\Omega} \frac{\partial \mathcal{F}}{\partial u} h + \frac{\partial \mathcal{F}}{\partial u'} h' dx$$

Integrition by parts $\int_a^b u dv = \underbrace{uv|_a^b}_{u(b)v(b) - u(a)v(a)} - \int_a^b v du$ over the second term

The Gâteaux Derivative

$$\left.\frac{dJ}{du}\right|_h = ... = \int_\Omega \frac{\partial \mathcal{F}}{\partial u} h + \frac{\partial \mathcal{F}}{\partial u'} h' dx$$

Integration by parts $\int_a^b u dv = \underbrace{uv|_a^b}_{u(b)v(b)-u(a)v(a)} - \int_a^b v du$ over the second term, choosing

$$u = \frac{\partial \mathcal{F}}{\partial u'} \qquad dv = h' dx$$

we have

$$du = \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} dx \qquad v = h$$

$$\left.\frac{dJ}{du}\right|_h = ... = \int_\Omega \left(\frac{\partial \mathcal{F}}{\partial u} h - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} h\right) dx + \left.\frac{\partial \mathcal{F}}{\partial u'} h\right|_a^b$$

$$= \int_\Omega \left(\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'}\right) h dx + \left.\frac{\partial \mathcal{F}}{\partial u'} h\right|_a^b = 0$$

The Gâteaux Derivative

$$\left.\frac{d\,J}{d\,u}\right|_h = ... = \int_\Omega \left(\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'}\right) h dx + \left.\frac{\partial \mathcal{F}}{\partial u'} h\right|_a^b = 0$$

We can impose (at the same time) two conditions

$$\begin{cases} 1) & \dfrac{\partial \mathcal{F}}{\partial u} - \dfrac{\partial}{\partial x}\dfrac{\partial \mathcal{F}}{\partial u'} = 0 \\ 2) & \left.\dfrac{\partial \mathcal{F}}{\partial u'} h\right|_a^b = 0 \end{cases}$$

Let us to study the second condition

The Gâteaux Derivative

Options on the boundary (boundary conditions):

1. $h(x)|_{\partial\Omega} = 0$, means that $u(x)|_{\partial\Omega}$ is fixed to some value. These are the **Dirichlet boundary conditions**. When $u(x)|_{\partial\Omega} = 0$, they are called homogeneous Dirichlet boundary conditions.
2. $h(x)|_{\partial\Omega} \neq 0$, means that one allows variations of $u(x)$ on the boundary, but then, following condition has to be fullfilled.

$$\left.\frac{\partial \mathcal{F}}{\partial u'}\right|_{\partial\Omega} = 0$$

These are called **Von Neumann boundary conditions**

(note: $\partial\Omega$ means boundary of Ω)

The Gâteaux Derivative

Let me remind you something

$$\left.\frac{d\,J(u)}{d\,u}\right|_h = \left\langle \frac{d\,J(u)}{d\,u}, h \right\rangle = \int \frac{d\,J(u)}{d\,u} h dx$$

And we aldeady prooved that

$$\left.\frac{d\,J(u)}{d\,u}\right|_h = \int_\Omega \left(\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} \right) h dx$$

So

$$\frac{d\,J(u)}{d\,u} = \left(\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} \right)$$

Euler-Lagrange Equation

For $\Omega \subset \mathbb{R}$
Dirichlet Boundary Conditions

$$\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} = 0 \qquad u(x)|_{\partial\Omega} = u_0(x)$$

Von Neumann Boundary Conditions

$$\frac{\partial \mathcal{F}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial \mathcal{F}}{\partial u'} = 0 \qquad \left.\frac{\partial \mathcal{F}}{\partial u'}\right|_{\partial\Omega} = 0$$

For $\Omega \subset \mathbb{R}^D$

$$\underbrace{\frac{\partial \mathcal{F}}{\partial u} - \sum_{i=1}^{D} \frac{\partial}{\partial x_i}\frac{\partial \mathcal{F}}{\partial u_{x_i}}}_{\frac{d\,J(u)}{d\,u}} = 0$$

where $\nabla u = (u_{x_1}, .., u_{x_D})$. The equation is completed with the corresponding boundary conditions

The Fundamental Problem of Variational Calculus

For a given energy functional

$$J(u) = \int_{\Omega} \mathcal{F}\left(x, u(x), \nabla u(x)\right) dx$$

We saw that the fundamental problem of the calculus of variations is to find the extremum (maximum or minimum) of the functional $J(u)$.

Necessary condition for the extremum

Under some assumptions over $\mathcal{F}$ and for the derivatives, if function u is a extremum of $J(u)$, then u is solution of the Euler-Lagrange (partial) differential equation

$$\frac{d\,J}{d\,u} = -\sum_{i=1}^{D} \frac{\partial}{\partial x_i} \frac{\partial \mathcal{F}}{\partial u_{x_i}} + \frac{\partial \mathcal{F}}{\partial u} = 0$$

where $\nabla u(x) = (u_{x_1}, ..u_{x_D})$ and D is the dimension .

Coming back to our previous example:

Denoising with the L^2 norm regularity term

Example

Denoising with the L^2 norm regularity term

The problem of denoising a given image is modelled as the minimization of

$$J(u) = \frac{1}{2}\int_\Omega |\nabla u|^2 dxdy + \frac{1}{2\lambda}\int_\Omega |u-f|^2 dxdy$$

Let's compute the Euler-Lagrange equation in this case. Afterwards, we can find the solution using gradient descent.

$$\begin{cases} u^{[0]} & = & u_0 \\ u^{[k+1]} & = & u^{[k]} - \tau \frac{d\,J(u)}{d\,u} \end{cases}$$

(stop when $(u^{[k+1]} - u^{[k]}) < \epsilon$)

Example

Denoising with the L^2 norm regularity term

Let's compute $\frac{d\,J(u)}{d\,u}$

$$\mathcal{F}(x, u(x), \nabla u(x)) = \frac{1}{2}|\nabla u|^2 + \frac{1}{2\lambda}|u - f|^2$$

Notice that $\nabla u = (u_x, u_y)$, so $\mathcal{F}$ can be written as

$$\mathcal{F}(x, u, \nabla u) = \frac{1}{2}\underbrace{\sqrt{u_x^2 + u_y^2}^2}_{u_x^2+u_y^2} + \frac{1}{2\lambda}|u - f|^2$$

Compute the needed elements

$$\frac{\partial \mathcal{F}}{\partial u} = \frac{1}{\cancel{2}\lambda}\cancel{2}(u - f)$$

$$\frac{\partial \mathcal{F}}{\partial u_x} = \frac{1}{\cancel{2}}\cancel{2}(u_x) \qquad \frac{\partial \mathcal{F}}{\partial u_y} = \frac{1}{\cancel{2}}\cancel{2}(u_y)$$

Example

Denoising with the L^2 norm regularity term

So

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - \left(\frac{\partial}{\partial x}u_x + \frac{\partial}{\partial y}u_y\right)$$

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - (\underbrace{u_{xx} + u_{yy}}_{\Delta u})$$

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - \Delta u$$

And the gradient descent scheme is

$$\begin{cases} u^{[0]} & = & u_0 \\ u^{[k+1]} & = & u^{[k]} - \tau\left(\frac{1}{\lambda}(u - f) - \Delta u\right) \end{cases}$$

Inconvenient: There are u without superscript.... Easy: Choose the superscript you want!!!!

Example:

Denoising with the L^2 norm regularity term

So

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - \left(\frac{\partial}{\partial x}u_x + \frac{\partial}{\partial y}u_y\right)$$

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - (\underbrace{u_{xx} + u_{yy}}_{\Delta u})$$

$$\frac{dJ(u)}{du} = \frac{1}{\lambda}(u - f) - \Delta u$$

And the gradient descent scheme is

$$\begin{cases} u^{[0]} & = & u_0 \\ u^{[k+1]} & = & u^{[k]} - \tau\left(\frac{1}{\lambda}(u - f) - \Delta u\right) \end{cases}$$

Inconvenient: There are u without superscript.... Easy: Choose the superscript you want: Explicit and Implicit algorithms

Choosing superscripts

Choose superscript for u (advice: $[k+1]$)

$$u^{[k+1]} = u^{[k]} - \tau \left(\frac{1}{\lambda}(u^{[k+1]} - f) - \Delta u \right)$$

$$u^{[k+1]} + \frac{\tau}{\lambda} u^{[k+1]} = u^{[k]} + \frac{\tau}{\lambda} f + \tau \Delta u$$

$$u^{[k+1]}(1 + \frac{\tau}{\lambda}) = u^{[k]} + \frac{\tau}{\lambda} f + \tau \Delta u$$

$$u^{[k+1]} = \frac{1}{(1 + \frac{\tau}{\lambda})} \left(u^{[k]} + \frac{\tau}{\lambda} f + \tau \Delta u \right)$$

Choosing superscripts

Choose superscript for Δu. Two possible choices (easy and hard)

- Easy: $[k]$. **Explicit methods** (everything on the right side is known)

$$u^{[k+1]} = \frac{1}{(1+\frac{\tau}{\lambda})}\left(u^{[k]} + \frac{\tau}{\lambda} f + \tau \Delta u^{[k]}\right)$$

- Hard: $[k+1]$. **Implicit methods.** You have to compute derivatives of some function you do not know!!

$$u^{[k+1]} = \frac{1}{(1+\frac{\tau}{\lambda})}\left(u^{[k]} + \frac{\tau}{\lambda} f + \tau \Delta u^{[k+1]}\right)$$

STOP!!!! Let's Recap

1. Model images as functions.
2. Model your image processing problem as a minimization of an energy functional.
3. Compute the derivative of the energy functional w.r.t the unknown.
4. Choose a (explicit or implicit) gradient descent method or solve the Euler-Lagrange equation.

Solving the Partial Differential Equation

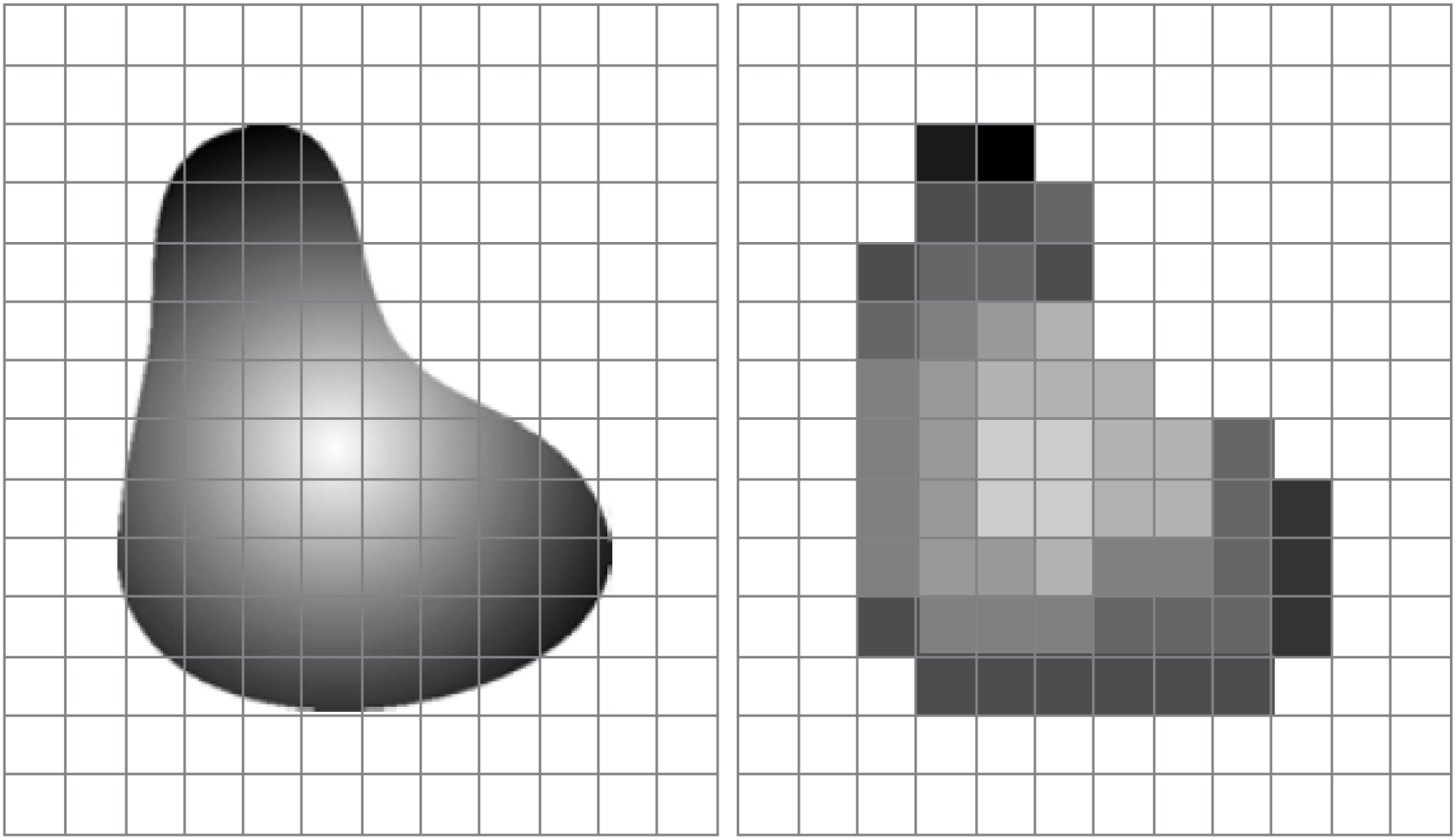
- We are not yet at the pixel level.
- How can we computationally solve the equation? Numerical Analysis

Numerical analysis is the study of algorithms that use numerical ***approximation*** (as opposed to general symbolic manipulations) for the problems of mathematical analysis (as distinguished from discrete mathematics).

wikipedia

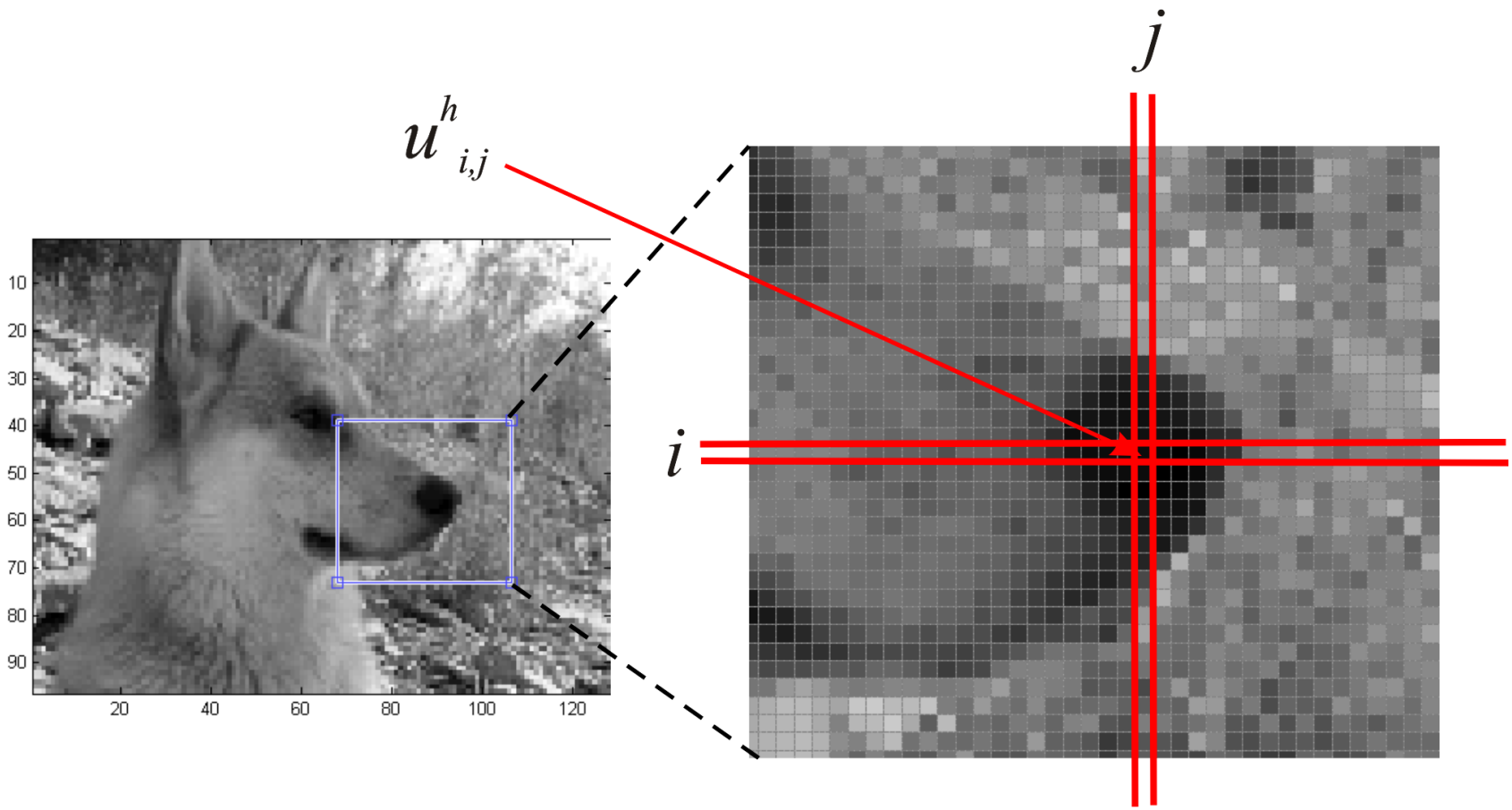
Discretization

Discretization concerns the process of transferring continuous functions, models, and equations into discrete counterparts



Discretization

Discretization concerns the process of transferring continuous functions, models, and equations into discrete counterparts



$$u(x, y) \approx u_{ij}$$

h: Distance between pixels.

Finite differences

We can approximate the derivative using Taylor as (there are many other possibilities)

$$\frac{\partial u(x,y)}{\partial x} \approx \frac{u(x+h,y)-u(x,y)}{h}$$

$$\frac{\partial u(x,y)}{\partial x} \approx \frac{u_{i+1,j}-u_{i,j}}{h}$$

$$\frac{\partial u(x,y)}{\partial y} \approx \frac{u_{i,j+1}-u_{i,j}}{h}$$

$$\frac{\partial^2 u(x,y)}{\partial x^2} \approx \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{h^2}$$

$$\frac{\partial^2 u(x,y)}{\partial y^2} \approx \frac{u_{ij+1}-2u_{ij}+u_{ij-1}}{h^2}$$

Finite differences

take h=1

Gradient Descent Example

Energy Functional

$$J(u) = \int_\Omega \frac{1}{2}|\nabla u|^2 + \frac{1}{2\lambda}|u - f|^2 dx$$

Rate of change w.r.t u

$$\frac{dJ}{du} = \frac{1}{\lambda}(u - f) - \Delta u$$

Gradient descent scheme

$$u^{[k+1]} = u^{[k]} - \tau\left(\frac{1}{\lambda}(u - f) - \Delta u\right)$$

$$u^{[k+1]} = \frac{1}{(1 + \frac{\tau}{\lambda})}\left(u^{[k]} + \frac{\tau}{\lambda}f + \tau\Delta u\right)$$

Explicit gradient descent scheme:

$$u_{i,j}^{[k+1]} = \frac{1}{(1 + \frac{\tau}{\lambda})}\left[u_{i,j}^{[k]} + \frac{\tau}{\lambda}f_{i,j} + \tau\left(u_{i+1,j}^{[k]} - 2u_{i,j}^{[k]} + u_{i-1,j}^{[k]} + u_{ij+1}^{[k]} - 2u_{ij}^{[k]} + u_{ij-1}^{[k]}\right)\right]$$

Implicit gradient descent scheme :

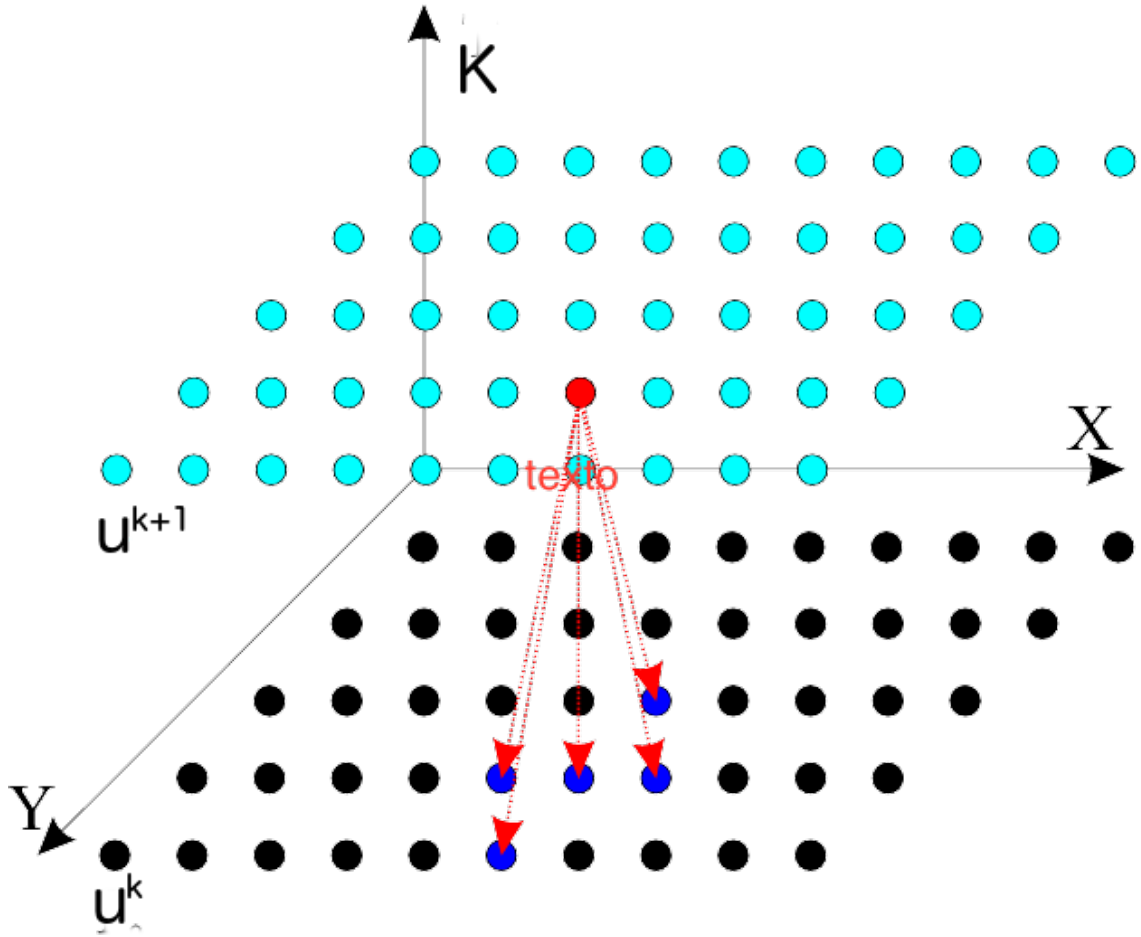
$$\left(4\tau + 1 + \frac{\tau}{\lambda}\right)u_{i,j}^{[k+1]} - \tau\left(u_{i+1,j}^{[k+1]} + u_{i-1,j}^{[k+1]} + u_{ij+1}^{[k+1]} + u_{ij-1}^{[k+1]}\right) = u_{i,j}^{[k]} + \frac{\tau}{\lambda}f_{ij}$$

Numerical Aspects

Gradient Descent

Explicit schemes

$$u_{i,j}^{[k+1]} = \frac{1}{(1+\frac{\tau}{\lambda})}\left[u_{i,j}^{[k]} + \frac{\tau}{\lambda} f_{i,j} + \tau\left(u_{i+1,j}^{[k]} - 2u_{i,j}^{[k]} + u_{i-1,j}^{[k]} + u_{ij+1}^{[k]} - 2u_{ij}^{[k]} + u_{ij-1}^{[k]}\right)\right]$$



Implicit schemes

Numerical Aspects

Implicit

With implicit methods, an algebraic system of equations $A\vec{x} = \vec{b}$ has to be solved where (usually A is sparse)

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

or leads to an algebraic system of non-linear equations $A(x) = b$

Numerical Aspects

Stationary Equation

Example:

$$\left(4\tau + 1 + \frac{\tau}{\lambda}\right) u_{i,j}^{[k+1]} - \tau \left(u_{i+1,j}^{[k+1]} + u_{i-1,j}^{[k+1]} + u_{ij+1}^{[k+1]} + u_{ij-1}^{[k+1]}\right) = u_{i,j}^{[k]} + \frac{\tau}{\lambda} f_{ij}$$

Ordering the pixels lexycographically by columns:

$$\begin{aligned} a_{ii} &= \left(4\tau + 1 + \tfrac{\tau}{\lambda}\right) \\ a_{i,i-1} &= -\tau \\ a_{i,i+11} &= -\tau \\ a_{i,i+n_x} &= -\tau \\ a_{i,i-n_x} &= -\tau \end{aligned}$$

$$\vec{x} = \begin{bmatrix} u_{11}^{[k+1]} \\ u_{21}^{[k+1]} \\ \vdots \\ u_{12}^{[k+1]} \\ u_{22}^{[k+1]} \\ \vdots \\ u_{n_x,n_y}^{[k+1]} \end{bmatrix}, \qquad b = \begin{bmatrix} u_{11}^{[k]} + \frac{\tau}{\lambda} f_{11} \\ u_{21}^{[k]} + \frac{\tau}{\lambda} f_{21} \\ \vdots \\ u_{12}^{[k]} + \frac{\tau}{\lambda} f_{12} \\ u_{22}^{[k]} + \frac{\tau}{\lambda} f_{22} \\ \vdots \\ u^{[k]} + \frac{\tau}{\lambda} f_{n_x,n_y} \end{bmatrix}$$

Being n_x and n_y the number of pixels on x and y directions.

Numerical Aspects

Gradient Descent

- Explicit gradient descent is very easy to implement.
- Explicit gradient descent is naturally parallel.
- Explicit gradient descent is conditionally stable, so τ has to be small $\rightarrow$ large number of iterations to converge.
- Implicit gradient descent use to be unconditionally stable so τ can be bigger $\rightarrow$ less number of iterations to converge.
- Implicit Gradient descent needs less iterations but an algebraic system of equations has to be solved.

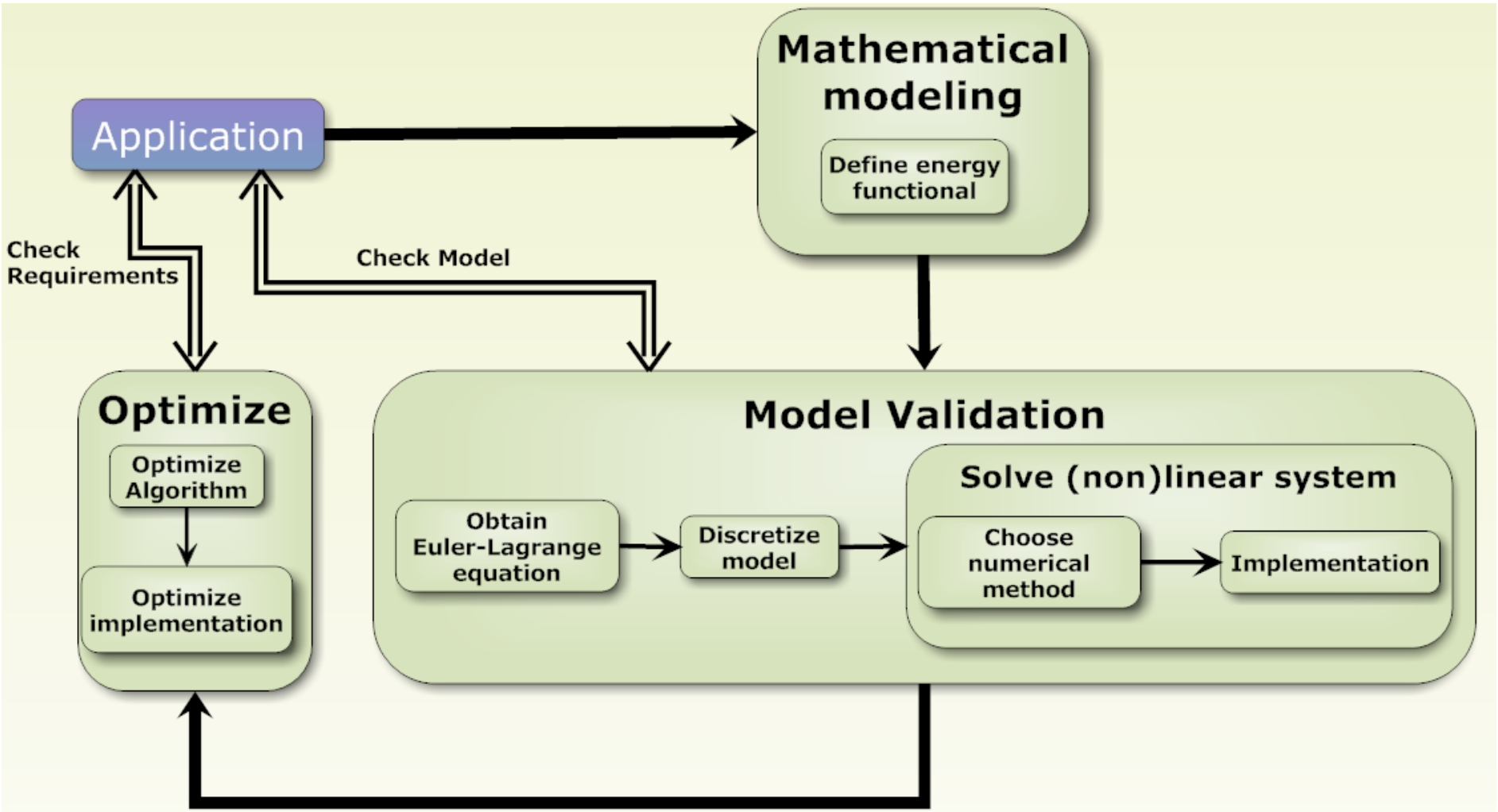
Boundary conditions

What happens on the boundary of the images? We do not have $u_{i-1,j}$, $u_{i+1,j}$, $u_{i,j-1}$, $u_{i,j+1}$ pixels!!!!!!!
We impose ANOTHER completely different condition, the boundary conditions:

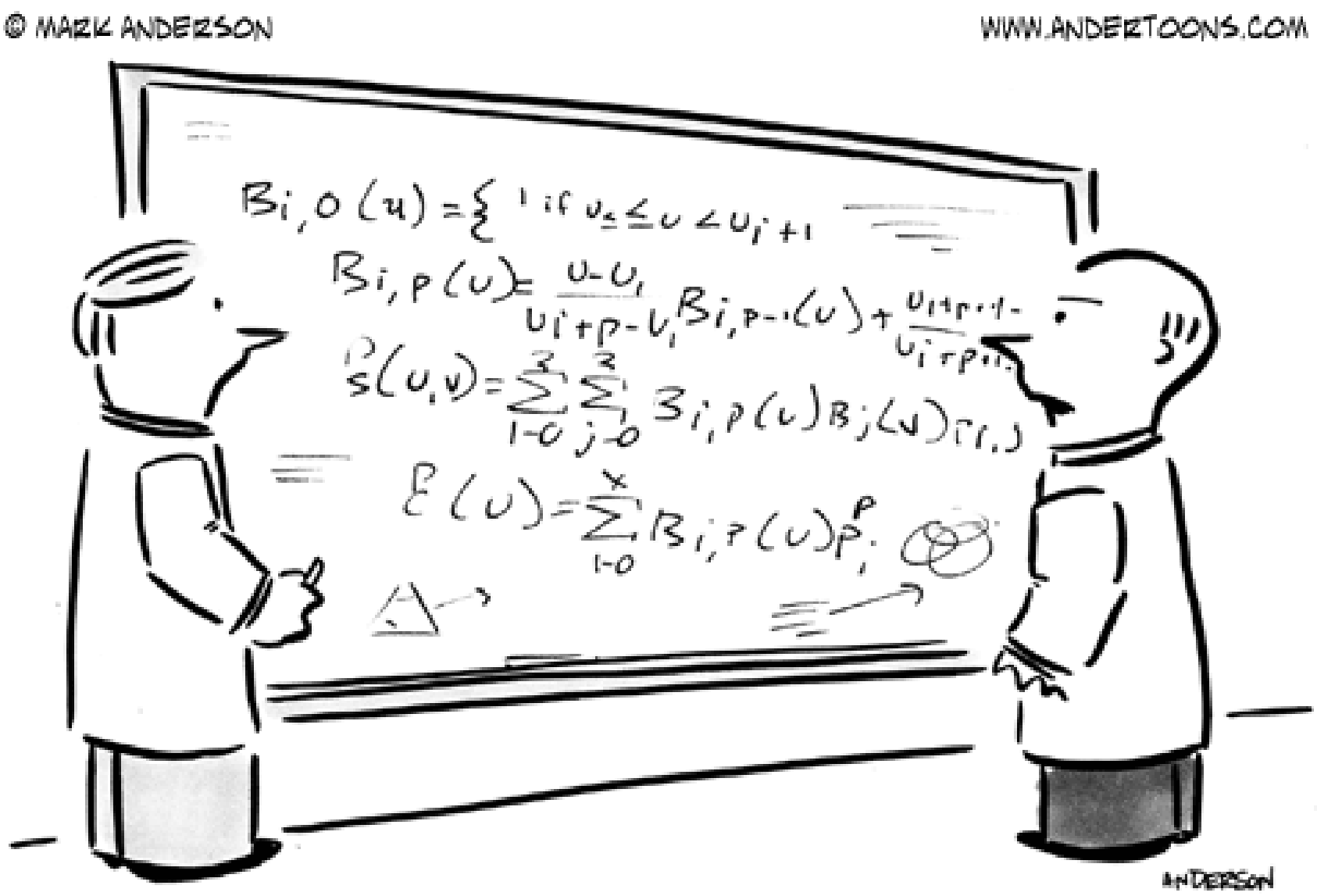
- Dirichlet: We know a priory the value of the **solution** u at the boundary, so we know the values for $u_{i-1,j}$, $u_{i+1,j}$, $u_{i,j-1}$, $u_{i,j+1}$
- Neumann: We know a priory the **rate of change** u at the boundary, so we know that

$$\left.\frac{\partial u}{\partial \bar{n}}\right|_{\partial\Omega} = \textit{some value}$$

Briefly


Application
Mathematical modeling
Define energy functional
Check Requirements
Check Model
Optimize
Optimize Algorithm
Optimize implementation
Model Validation
Obtain Euler-Lagrange equation
Discretize model
Solve (non)linear system
Choose numerical method
Implementation

Questions



"What the hell is *that* supposed to mean?!"