



# Chapter 3

## Variational Calculus

Computer Vision I: Variational Methods

Winter 2016/17

Variational Methods

Image Denoising

Iterative Solvers

Infinite-Dimensional  
Setting

The Gâteaux  
Derivative

The Euler-Lagrange  
Equation

Gradient Descent

Boundary Conditions

Diffusion as Gradient  
Descent

Euler and Lagrange

Prof. Daniel Cremers  
Chair for Computer Vision and Pattern Recognition  
Departments of Informatics & Mathematics  
Technical University of Munich

- 1 Variational Methods
- 2 Image Denoising
- 3 Iterative Solvers
- 4 Infinite-Dimensional Setting
- 5 The Gâteaux Derivative
- 6 The Euler-Lagrange Equation
- 7 Gradient Descent
- 8 Boundary Conditions
- 9 Diffusion as Gradient Descent
- 10 Euler and Lagrange



Variational Methods

Image Denoising

Iterative Solvers

Infinite-Dimensional  
Setting

The Gâteaux  
Derivative

The Euler-Lagrange  
Equation

Gradient Descent

Boundary Conditions

Diffusion as Gradient  
Descent

Euler and Lagrange

## Variational Methods

**Variational methods** are a specific class of optimization methods. The key idea is to define cost functionals over a **continuous solution space** and to compute optima by solving the corresponding **extremality principle**.

Variational methods allow to solve respective problems in a **mathematically transparent** manner. Instead of performing a heuristic sequence of processing steps one starts by defining what properties a solution should have. Once these are fixed, the appropriate algorithm can be derived “automatically”.

Variational methods are particularly suited for infinite-dimensional problems. They are among the **top performing methods** for:

- image denoising, deblurring, super-resolution
- image segmentation
- motion estimation
- dense 3D reconstruction
- tracking



## Advantages of Optimization Methods

Optimization methods have many advantages over traditional multi-step approaches (such as the Canny edge detector):

- A mathematical analysis of the cost function allows statements regarding the **existence**, **uniqueness** and **stability** of solutions to a given problem.
- In **traditional multistep processes** the interplay of consecutive steps is often **complex** and **intransparent**. It is typically unclear how modifying or replacing one component affects the subsequent steps.
- Optimization methods are based on **transparent and explicitly formulated assumptions**, with no “hidden” assumptions.
- In general, optimization methods have **fewer parameters**. The meaning of each parameter is fairly obvious.
- Optimization methods are **easily combined** in a transparent manner (by adding respective cost functions).



## A Simple Example: Image Denoising

Let  $f : \Omega \rightarrow \mathbb{R}$  be an input image corrupted by noise. The goal is to compute a **denoised version  $u$  of the image  $f$** .

The desired function  $u$  should fulfill two criteria:

- The function  $u$  should be **similar to  $f$** .
- The function  $u$  should be **spatially smooth**.

Both criteria can be combined in the following cost function (or energy):

$$E(u) = E_{data}(u, f) + \lambda E_{smoothness}(u),$$

where the first term measures the similarity of  $u$  and  $f$  and the second term measures the smoothness of  $u$ . A weighting or **regularization parameter  $\lambda \geq 0$**  specifies the relative importance of smoothness versus data fit.

Most variational approaches have the above form. They merely differ in how the similarity term (**data term**) and the smoothness term (**regularizer**) are defined.



## Discrete Denoising in 1D

Let us assume for now that  $f$  is discrete in one spatial variable, i.e.  $f = \{f_1, f_2, \dots, f_n\}$  is simply a sequence of brightness values  $f_i \in \mathbb{R}$ . We seek an approximation  $u = \{u_1, \dots, u_n\}$ .

The data term which measures similarity of  $f$  and  $u$  can for example be written as:

$$E_{data}(u) = \frac{1}{2} \sum_{i=1}^n (f_i - u_i)^2,$$

which means that we measure the overall brightness difference as a **sum of squared differences (SSD)**.

The smoothness term can for example be written as:

$$E_{smooth}(u) = \frac{1}{2} \sum_{i=1}^{n-1} (u_i - u_{i+1})^2,$$

which means that we measure the sum of squared differences for all neighboring brightness values.



The total energy is thus:

$$E_\lambda(u) = \frac{1}{2} \sum_{i=1}^n (f_i - u_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{n-1} (u_i - u_{i+1})^2,$$

Larger values of  $\lambda$  imply that the smoothness of the solution should play a bigger role.

A solution to the above denoising problem is a function  $\hat{u}$  which minimizes the above energy:

$$\hat{u} = \arg \min_u E_\lambda(u).$$

Variational methods determine functions which fulfill the **extremality principle**:

$$\frac{dE_\lambda(u)}{du} = 0, \quad \Leftrightarrow \quad \frac{\partial E_\lambda(u)}{\partial u_i} = 0 \quad \forall i \in [1, n]$$



## Discrete Denoising in 1D

The extremality condition for each pixel is therefore:

$$\frac{\partial E_\lambda(u)}{\partial u_1} = (u_1 - f_1) + \lambda(u_1 - u_2) = 0,$$

$$\frac{\partial E_\lambda(u)}{\partial u_i} = (u_i - f_i) + \lambda(2u_i - u_{i-1} - u_{i+1}) = 0, \quad \forall i \in [2, n-1],$$

$$\frac{\partial E_\lambda(u)}{\partial u_n} = (u_n - f_n) + \lambda(u_n - u_{n-1}) = 0.$$

These conditions form a **system of linear equations**:

$$M_\lambda u = \begin{pmatrix} 1+\lambda & -\lambda & & & & \\ -\lambda & 1+2\lambda & -\lambda & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\lambda & 1+2\lambda & -\lambda & \\ & & & -\lambda & 1+\lambda & \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix}$$







- The matrix  $M_\lambda$  is tridiagonal.
- The inverse matrix  $M_\lambda^{-1}$  exists. It is dense and has only non-negative entries.
- The above system of equations can be solved in linear time using Gaussian elimination (Thomas' algorithm).
- The minimizer is **unique** because  $E(u)$  is strictly convex.
- Reminder:
  - A **set** is called **convex**, if for any pair of points in the set, the connecting line is also contained in the set.
  - A **function**  $E(u)$  is called **convex**, if its epigraph is convex, i.e. if for any two points on the graph of  $E$  the function values are below the connecting line.
  - $E(u)$  is called **strictly convex** (streng konvex) if the function values are strictly below the connecting line:

$$E((1-\alpha)u_1 + \alpha u_2) < (1-\alpha)E(u_1) + \alpha E(u_2) \quad \forall u_1, u_2 \quad \forall \alpha \in (0, 1).$$

## Proof of convexity

The function  $h(s) = s^2$  is convex:

$$h((1-\alpha)s_1 + \alpha s_2) < (1-\alpha)h(s_1) + \alpha h(s_2), \quad \forall s_1, s_2, \forall \alpha \in (0, 1).$$

For any  $u = (u_1, \dots, u_n)$  and  $\tilde{u} = (\tilde{u}_1, \dots, \tilde{u}_n)$  and for any  $\alpha \in (0, 1)$  we therefore have:

$$\begin{aligned} E_\lambda((1-\alpha)u + \alpha\tilde{u}) &= \frac{1}{2} \sum_{i=1}^n \left( f_i - ((1-\alpha)u_i + \alpha\tilde{u}_i) \right)^2 \\ &+ \frac{\lambda}{2} \sum_{i=1}^{n-1} \left( ((1-\alpha)u_i + \alpha\tilde{u}_i) - ((1-\alpha)u_{i+1} + \alpha\tilde{u}_{i+1}) \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^n \left( (1-\alpha)(f_i - u_i) + \alpha(f_i - \tilde{u}_i) \right)^2 \\ &+ \frac{\lambda}{2} \sum_{i=1}^{n-1} \left( ((1-\alpha)(u_i - u_{i+1}) + \alpha(\tilde{u}_i - \tilde{u}_{i+1})) \right)^2 \\ &< (1-\alpha)E_\lambda(u) + \alpha E_\lambda(\tilde{u}). \end{aligned}$$



## Discrete Denoising ( $d$ -dim.)

Index all pixels of the  $d$ -dim volume with index  $i \in [1, \dots, N]$ , where  $N = n_1 \cdot n_2 \cdots n_d$ .

Variational denoising of an image  $f$ :

$$E_\lambda(u) = \frac{1}{2} \sum_{i=1}^N (f_i - u_i)^2 + \frac{\lambda}{2} \sum_{i=1}^N \sum_{j \in \mathcal{N}(i)} (u_i - u_j)^2,$$

where  $\mathcal{N}(i)$  denotes the **neighborhood of pixel  $i$** .

Again  $E$  is strictly convex. The condition for (global) optimality is:

$$\frac{dE_\lambda(u)}{du_i} = (u_i - f_i) + \lambda \sum_{j \in \mathcal{N}(i)} (u_i - u_j) = 0 \quad \forall i$$

In the higher-dimensional case, this gives rise to a large-scale linear programming problem.





For dimensions ( $d = 2, 3, \dots$ ), the linear equation system

$$M_\lambda u = f$$

is quite large.

Most entries of the matrix  $M_\lambda$  are 0 (**sparse matrix**). Yet, its inverse is typically difficult to compute or even to store in memory.

There exist numerous standard solvers for large linear systems. The best known ones are the **Jacobi method** and the **Gauss-Seidel method**.

In the following, we will discuss the Jacobi solver, the Gauss-Seidel solver and various extensions.

## The Jacobi Method

The Jacobi method converges if the matrix  $M \equiv M_\lambda$  is **strictly diagonally dominant**, i.e. if for any row of the matrix the absolute value of the diagonal element is larger than the sum of absolute values of the off-diagonal elements:

$$|m_{ii}| > \sum_{j \neq i} |m_{ij}| \quad \forall i$$

Decompose the matrix  $M = D + A$  into its **diagonal part  $D$**  and the **offdiagonal part  $A$** . Then:

$$Mu = (D + A)u = f \quad \Leftrightarrow \quad Du = f - Au$$

Initialize with an arbitrary function  $u^0$  and iterate:

$$u^{(k+1)} = D^{-1}(f - Au^{(k)}), \quad k = 0, 1, 2, 3, \dots$$

Low computational and memory cost, easily parallelizable.

We can use the residuum  $r^{(k)} := Mu^{(k)} - f$  as a **termination criterion**:  $\frac{|r^{(k)}|}{|r^{(0)}|} < \epsilon$ .





- Separate diagonal part  $D$ :

$$u_i + \lambda |\mathcal{N}(i)| u_i = f_i + \lambda \sum_{j \in \mathcal{N}(i)} u_j \quad \forall i,$$

where  $\mathcal{N}(i)$  is the number of neighbors of pixel  $i$ .

- Iteration:

$$u_i^{(k+1)} = \frac{f_i + \lambda \sum_{j \in \mathcal{N}(i)} u_j^{(k)}}{1 + \lambda |\mathcal{N}(i)|}$$

- Appropriate initialization:

$$u^{(0)} = f$$

- **Intuitive Interpretation:** For each pixel  $i$ , the above iteration induces a weighted averaging of brightness values in its neighborhood  $\mathcal{N}(i)$ . The weights sum to 1.

## Alternative Methods

Besides the Jacobi method, there are other methods for solving linear equation systems of the form  $Mu = f$ . In particular:

- **Gauss-Seidel method:** Updates pixel values sequentially always using the most recent values:

$$u_i^{(k+1)} = \frac{1}{m_{ii}} \left( f_i - \sum_{j < i} m_{ij} u_j^{(k+1)} - \sum_{j > i} m_{ij} u_j^{(k)} \right)$$

Faster than the Jacobi method. But: not parallelizable, solution depends on the order of updates.

- **SOR method (Successive Over-Relaxation):** Extrapolate the Gauss-Seidel updates linearly for faster convergence:

$$u_i^{(k+1)} = \omega \bar{u}_i^{(k+1)} + (1 - \omega) u_i^{(k)}$$

where  $\bar{u}_i^{(k+1)}$  is the current solution of the Gauss-Seidel method, and  $\omega \in [1, 2)$  the extrapolation factor ( $\omega = 1$ : Gauss-Seidel).





There exists a multitude of numerical strategies to accelerate algorithms. Two common examples are:

- **Preconditioned Conjugate Gradients:** Modification of the gradient descent or steepest descent which minimizes along orthogonal directions. Preconditioning can additionally improve the convergence rate.
- **Multigrid Methods:** Solve the equation system starting from a coarse-grid representation and use solution as initialization on respective finer grids.

Which combination of methods leads to the fastest solution depends on the type of optimization problem / cost function and the available hardware.

For example, the Gauss-Seidel and SOR methods are typically faster than the Jacobi method, but they are not directly parallelizable.





A spatially continuous variational approach to image denoising and restoration looks as follows:

Given an image  $f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$  find a smooth approximation  $u : \Omega \rightarrow \mathbb{R}$  of this image. This can be determined by minimizing a cost function of the form:

$$E(u) = \frac{1}{2} \int (u(x) - f(x))^2 dx + \frac{\lambda}{2} \int |\nabla u(x)|^2 dx$$

This is the spatially continuous analogue of the previously discussed discrete formulation.

Such a mapping which assigns a real number  $E(u)$  to a function  $u(x)$  is also referred to as a **functional**.

How does one minimize the above functional with respect to the function  $u$ ?

[Variational Methods](#)[Image Denoising](#)[Iterative Solvers](#)[Infinite-Dimensional Setting](#)[The Gâteaux Derivative](#)[The Euler-Lagrange Equation](#)[Gradient Descent](#)[Boundary Conditions](#)[Diffusion as Gradient Descent](#)[Euler and Lagrange](#)



A **functional** is a mapping  $E$  which assigns to each element of a vector-space (to each function  $u$ ) an element from the underlying field (a number).

Let

$$E(u) = \int \mathcal{L}(u, u') dx$$

be a functional, where  $u' = \frac{du}{dx}$  is the derivative of the function  $u$ . (In physics  $\mathcal{L}$  is called the **Lagrange density**).

**Example:**  $\mathcal{L}(u, u') = \frac{1}{2}(u(x) - f(x))^2 + \frac{\lambda}{2}|u'(x)|^2$ .

Just as with real-valued functions defined on  $\mathbb{R}^n$  the necessary condition for extremality of the functional  $E$  states that the **derivative with respect to  $u$  must be 0**.

Yet how does one define and compute the derivative of a functional  $E(u)$  with respect to the function  $u$ ?

## The Gâteaux Derivative

There are several ways to introduce functional derivatives. The following definition goes back to works of the French mathematician **R. Gâteaux** († 1914) which were published posthumously in 1919: [http:](http://archive.numdam.org/ARCHIVE/BSMF/BSMF_1919__47_/BSMF_1919__47__47_1/BSMF_1919__47__47_1.pdf)

[//archive.numdam.org/ARCHIVE/BSMF/BSMF\\_1919\\_\\_47\\_/BSMF\\_1919\\_\\_47\\_\\_47\\_1/BSMF\\_1919\\_\\_47\\_\\_47\\_1.pdf](http://archive.numdam.org/ARCHIVE/BSMF/BSMF_1919__47_/BSMF_1919__47__47_1/BSMF_1919__47__47_1.pdf)

The Gâteaux derivative extends the concept of directional derivative to infinite-dimensional spaces.

The derivative of the functional  $E(u)$  in direction  $h(x)$  is defined as:

$$\left. \frac{dE(u)}{du} \right|_h = \lim_{\epsilon \rightarrow 0} \frac{E(u + \epsilon h) - E(u)}{\epsilon}$$

As in finite dimensions, this **directional derivative** can be interpreted as the **projection of the functional gradient on the respective direction**. We can therefore write:

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



## The Gâteaux Derivative

For functionals of the **canonical form**:  $E(u) = \int \mathcal{L}(u, u') dx$  the Gâteaux derivative is given by

$$\begin{aligned} \left. \frac{dE(u)}{du} \right|_h &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} (E(u + \epsilon h) - E(u)) \\ &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \int (\mathcal{L}(u + \epsilon h, u' + \epsilon h') - \mathcal{L}(u, u')) dx \\ &= \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \int \left( \left( \mathcal{L}(u, u') + \frac{\partial \mathcal{L}}{\partial u} \epsilon h + \frac{\partial \mathcal{L}}{\partial u'} \epsilon h' + o(\epsilon^2) \right) - \mathcal{L}(u, u') \right) dx \\ &= \int \left( \frac{\partial \mathcal{L}}{\partial u} h + \frac{\partial \mathcal{L}}{\partial u'} h' \right) dx \\ &= \int \left( \frac{\partial \mathcal{L}}{\partial u} h - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} h \right) dx \quad (\text{partial int., } h = 0 \text{ on boundary}) \\ &= \int \underbrace{\left( \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} \right)}_{\frac{dE}{du}} h(x) dx. \end{aligned}$$



## Euler-Lagrange Equation

Thus the derivative of the functional  $E(u)$  in direction  $h$  is:

$$\frac{dE(u)}{du} \Big|_h = \int \underbrace{\left( \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} \right)}_{\frac{dE}{du}} h(x) dx.$$

As a necessary condition for minimality of the functional  $E(u)$  the **variation of  $E$  in any direction  $h(x)$  must vanish**. Therefore at the extremum we have:

$$\boxed{\frac{dE}{du} = \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} = 0}$$

This condition is called the **Euler-Lagrange equation**.

**Example:** For  $\mathcal{L}(u, u') = \frac{1}{2}(u(x) - f(x))^2 + \frac{\lambda}{2}|u'(x)|^2$ , we get:

$$\frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} = (u(x) - f(x)) - \frac{d}{dx}(\lambda u'(x)) = u - f - \lambda u'' = 0$$



# Functional Minimization

The Euler-Lagrange equation is a differential equation which forms the **necessary condition for minimality**.

The central idea of variational methods is to compute solutions to the respective Euler-Lagrange equation.

This can be done in several ways. For example, one can discretize the function  $u$  on a set of points  $\{x_1, \dots, x_n\}$  and subsequently try to solve for the values  $u(x_i)$ . For quadratic cost functions, the arising set of linear equations can be solved using the discussed iterative algorithms (Jacobi, Gauss-Seidel,...). In general, however, the Euler-Lagrange equation will not be linear.

For general (non-quadratic) energies, one can start with an initial guess  $u_0(x)$  of the solution and iteratively improve the solution. Such methods are called **descent methods**.

How can one iteratively improve a given solution?



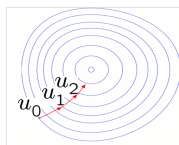
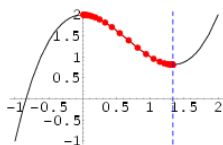
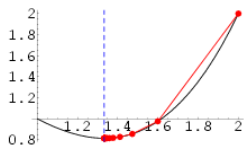
# Gradient Descent

**Gradient descent** or steepest descent is a particular descent method where in each iteration one chooses the direction in which the energy decreases most. The **direction of steepest descent** is given by the **negative energy gradient**.

To minimize a real-valued function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , the gradient descent for  $f(u)$  is defined by the differential equation:

$$\begin{cases} u(0) = u_0 \\ \frac{du}{dt} = -\frac{df}{du}(u) \end{cases}$$

Discretization:  $u_{t+1} = u_t - \epsilon \frac{df}{du}(u_t), \quad t = 0, 1, 2, \dots$



## Gradient Descent

For minimizing functionals  $E(u)$ , the gradient descent is done analogously.

For the functional  $E(u) = \int \mathcal{L}(u, u') dx$ , the gradient is given by:

$$\frac{dE}{du} = \frac{d\mathcal{L}}{du} - \frac{d}{dx} \frac{d\mathcal{L}}{du'}$$

Therefore the gradient descent is given by:

$$\begin{cases} u(x, 0) = u_0(x) \\ \frac{\partial u(x, t)}{\partial t} = -\frac{dE}{du} = -\frac{d\mathcal{L}}{du} + \frac{\partial}{\partial x} \frac{d\mathcal{L}}{du'} \end{cases}$$

For  $\mathcal{L}(u, u') = \frac{1}{2}(u - f)^2 + \frac{\lambda}{2}|u'|^2$ , this means:

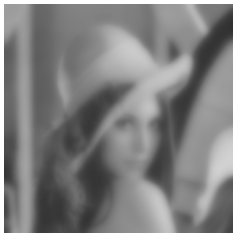
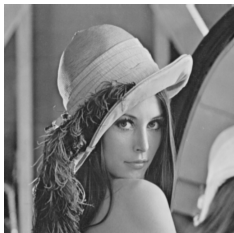
$$\frac{\partial u}{\partial t} = (f - u) + \lambda u'' = (f - u) + \lambda \Delta u.$$

If the gradient descent converges, i.e.  $\partial_t u = -\frac{dE}{du} = 0$ , then we have found a solution to the Euler-Lagrange equation.

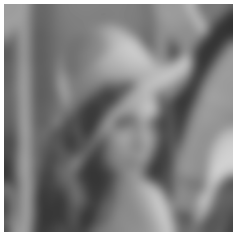
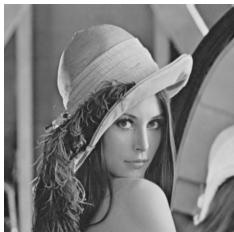




## Diffusion with Data Term



$$E(u) = \int (f - u)^2 dx + \lambda \int |\nabla u|^2 dx \rightarrow \min.$$



$$E(u) = \lambda \int |\nabla u|^2 dx \rightarrow \min.$$

Author: D. Cremers



## Addendum: Boundary Conditions

When deriving the Euler-Lagrange equations we only considered perturbations  $h(x)$  which are 0 on the boundary.

Without this assumption, Gâteaux's directional derivative is:

$$\begin{aligned}\frac{dE(u)}{du}\Big|_h &= \dots = \int_a^b \left( \frac{\partial \mathcal{L}}{\partial u} h + \frac{\partial \mathcal{L}}{\partial u'} h' \right) dx \\ &= \int_a^b \left( \frac{\partial \mathcal{L}}{\partial u} h - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} h \right) dx + \left( \frac{\partial \mathcal{L}}{\partial u'} h(x) \right)_a^b \\ &= \int_a^b \left( \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} \right) h(x) dx + \left( \frac{\partial \mathcal{L}}{\partial u'} h(x) \right)_a^b = 0\end{aligned}$$

$$\Rightarrow \begin{cases} 1) & \frac{dE}{du} = \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} = 0 \\ 2) & \left( \frac{\partial \mathcal{L}}{\partial u'} h(x) \right)_a^b = 0 \end{cases}$$





## Addendum: Boundary Conditions

Depending on the application one can distinguish two kinds of boundary conditions:

- **Dirichlet boundary conditions:** The function  $u(x)$  is fixed on the boundary ( $u_r(x)$ ), i.e.  $h(x) = 0$  on the boundary. One only considers variations of  $u(x)$  inside the domain:

$$\begin{cases} \frac{dE}{du} = \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} = 0 \\ u(x) \Big|_{\text{boundary}} = u_r(x) \end{cases}$$

- **Von Neumann boundary conditions:** One additionally allows for variations of  $u(x)$  on the boundary:

$$\begin{cases} \frac{dE}{du} = \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx} \frac{\partial \mathcal{L}}{\partial u'} = 0 \\ \frac{\partial \mathcal{L}}{\partial u'} \Big|_{\text{boundary}} = 0 \end{cases}$$

Many **diffusion equations** (albeit not all) can be derived as the **gradient descent** on a specific energy.

The energy:

$$E(u) = \int \mathcal{L}(u, \nabla u) dx = \frac{1}{2} \int g(x) |\nabla u(x)|^2 dx$$

leads to the gradient descent:

$$\frac{\partial u(x, t)}{\partial t} = -\frac{dE}{du} = -\frac{\partial \mathcal{L}}{\partial u} + \nabla \frac{\partial \mathcal{L}}{\partial \nabla u} = \nabla \left( g(x) \nabla u \right)$$

This equation corresponds to an **inhomogeneous diffusion** with diffusivity  $g(x)$ .

In other words, the above inhomogeneous diffusion process is nothing but a steepest descent on the weighted smoothness energy.



Variational Methods

Image Denoising

Iterative Solvers

Infinite-Dimensional Setting

The Gâteaux Derivative

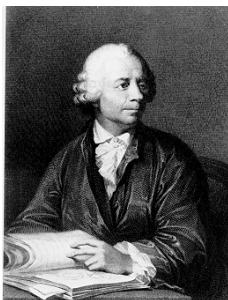
The Euler-Lagrange Equation

Gradient Descent

Boundary Conditions

Diffusion as Gradient Descent

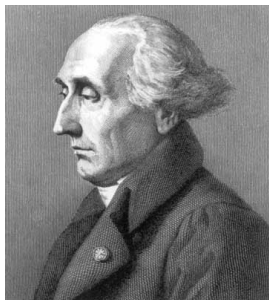
Euler and Lagrange



Leonhard Euler (1707 – 1783)

- Published 886 papers and books, most of them in the last 20 years of his life. Considered the greatest mathematician of the 18th century.
- Major contributions: Euler number  $e$ , Euler angles, Euler formula, Euler theorem, Euler equations (for fluid flows), Euler-Lagrange equations,...
- 13 children





Joseph-Louis Lagrange (1736 – 1813)

- born Giuseppe Lodovico Lagrangia (in Turin). self-taught.
- With 17 years: Professor for mathematics in Turin.
- Later in Berlin (1766-1787) and Paris (1787-1813).
- 1788: *La Mécanique Analytique*.
- 1800: *Leçons sur le calcul des fonctions*.