INSA Toulouse

Department of Applied Mathematics

Data Assimilation

Inverse Problems, Variational Assimilation, Optimal Control, Learning

by Jérôme Monnier

NB. This document is still in preparation. Consequently, it contains some personal annotations, and there may be missing paragraphs which, however, are not part of the course program.

Goals: to solve inverse problems by data assimilation into physically-based models, to infer, identify model parameters, to calibrate models, to compute model output gradients of large dimension

Keywords: inverse problems, best estimates, optimisation, gradient-based algorithms, model-based feedback control, PDE-based models, adjoint method, deep learning, PINNs.

Required knowledge: basic approximation methods, differential calculus, numerical optimisation, classical PDE models (weak forms and basics of functional analysis are a plus), numerical schemes (Finite Differences, Finite Element is a plus), artificial neural networks, Python programming.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

INSTITUT DE MATHÉMATIQUES de TOULOUSE

Université de Toulouse

Contents

III Variational Approaches for ODEs and General PDEs-based models 93

[1]

[1] The sections indicated with a * are "to go further sections". These sections can be skipped as a first reading, or if you are not interested in deeper mathematical basis, mathematical proofs.

WhereTo & How

Goals of this course
- To review and deepen understanding of fundamental numerical methods for analyzing and solving inverse problems.
-To learn the basics of traditional Data Assimilation methods (sequential/filters, variational) and Bayesian analysis, as well as their connections.
- To learn the fundamentals of Physically Informed Neural Network (PINNs) models and their connections with Data Assimilation.
- To study the Variational Data Assimilation (VDA) method in detail.
- To learn the basics of optimal control for systems governed by Partial Differential Equations (PDEs) or Ordinary Differential Equations (ODEs).
- To compute gradients of large-dimensional model outputs using the adjoint method.
- To learn how to assess computational codes, including adjoint codes and resulting gradients.
- To address real-world inverse problems by optimally combining mathematical-physical equations (primarily PDE models), databases containing measurements of phenomena, and probabilistic priors.
- To perform model calibration, parameter identification, and local sensitivity analysis by assimilating data into the model.

At the end, the students are supposed to be able :
- To set up and implement a data assimilation formulation in Python for fields such as geophysical fluid mechanics, structural mechanics, biology, etc., using given databases.
- To compute large-dimensional gradients by deriving the adjoint model and designing the complete optimization process.
- To perform model calibration and estimate uncertain parameters by assimilating data into the physics-based model.

Content Please consult the table of contents.
Numerous **Python codes** are provided to illustrate methods for solving inverse problems.

On the operation of this training program: please consult the INSA Moodle page of the course.

Part I

Inverse Problems: Basics Principles and Tools, Examples

Chapter 1

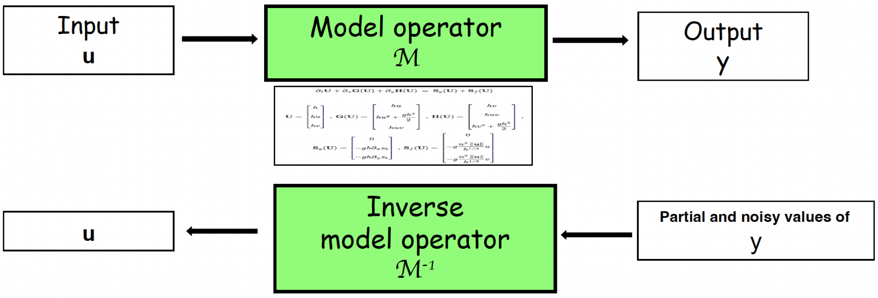
Inverse problems

The outline of this chapter is as follows.

Contents

1.1 Direct - inverse?

In the common sense, the term "model" denotes a *direct* model (also called *"forward"* model).



Input
u
Model operator
$\mathcal{M}$
Output
y
$\partial_t \mathbf{U} + \partial_x \mathbf{G}(\mathbf{U}) + \partial_y \mathbf{H}(\mathbf{U}) = \mathbf{S}_g(\mathbf{U}) + \mathbf{S}_f(\mathbf{U})$
$\mathbf{U} = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{gh^2}{2} \\ huv \end{bmatrix}, \mathbf{H}(\mathbf{U}) = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{gh^2}{2} \end{bmatrix},$
$\mathbf{S}_g(\mathbf{U}) = \begin{bmatrix} 0 \\ -gh\partial_x z_b \\ -gh\partial_y z_b \end{bmatrix}, \mathbf{S}_f(\mathbf{U}) = \begin{bmatrix} 0 \\ -g\frac{n^2\|\mathbf{u}\|}{h^{1/3}}u \\ -g\frac{n^2\|\mathbf{u}\|}{h^{1/3}}v \end{bmatrix}$
u
Inverse
model operator
$\mathcal{M}^{-1}$
Partial and noisy values of
y

Figure 1.1: Representation of a direct model with its input variable ("parameter") u (a-priori vectorial) and its output variable y; and its inverse counterpart.

A direct problem based on the model operator $\mathcal{M}(.)$ consists to find (compute) a solution y given the input parameter u: find y, $y = \mathcal{M}(u)$.

The inverse problem consists to find u, given y i.e. to compute $u = \mathcal{M}^{-1}(y)$.

Numerous excellent books have been published on inverse problems, let us cite for example [42, 38, 23, 29, 36, 4].

1.2 Examples

Many examples are presented e.g. in [18, 15, 29, 27].

Example 1) The historical Lagrange interpolation problem.
Find a polynomial $p(x)$ of degree n, of coefficients $(u_0, \cdots, u_n)$ that fit given values $(y_1, , ..., y_n)$ at some points $(x_1, ..., x_n)$.
This Lagrange interpolation problem is actually the inverse problem of the following direct problem: calculate the given polynomial $p(x)$ at the points $(x_1, ..., x_n)$.

This example is somehow a problem of *parameters identification*, given the "model" $p(u; x)$.

Example 2) A PDE-based identification problem.
Let us consider a diffusion phenomena in a material. The non homogeneous diffusivity of the material (e.g. a conductivity of a biological tissue) is denoted by $u(x)$.
The model is the following.
Given $u(x)$ in the domain Ω, find the scalar quantity $y(x,t)$ (e.g. an electrical field or wave intensity) satisfying:

$$\begin{cases} \partial_t y(x,t) - div(u(x)\nabla y(x,t)) = 0 & \text{in } \Omega\times]0,T[\\ y(x,0) \ = \ y_0(x) & \text{in } \Omega \\ y(x,t) = y_d(x,t) & \text{in } \partial\Omega\times]0,T[\end{cases} \tag{1.1}$$

The initial condition y_0 and the value y_d at boundary are given.
The direct problem consists to solve this classical Boundary Value Problem (BVP).
It is a well-posed problem (see later for exact meaning).

The inverse problem is as follows.
Given some *boundary measurements* of the field $y(x,t)$ and the flux $[u(x)\partial_n y(x,t)]$ on $\partial\Omega$, determine the unknown / uncertain diffusivity coefficient $u(x)$ in the domain Ω.

The Electrical Impedance Tomography (EIT) problem This inverse problem described above corresponds to the impedance tomography problem. A particular case is the Electrical Impedance Tomography (EIT) problem.
" Electrical Impedance Tomography (EIT) is a noninvasive type of medical imaging in which the electrical conductivity, permittivity, and impedance of a part of the body is inferred from surface electrode measurements and used to form a tomographic image of that part" (from Wikipedia page). See eg. Fig. 1.2.
In the context of Electrical Impedance Tomography (EIT), the inverse problem aims at recovering conductivity (and permittivity) inside a body from surface measurements of electrical currents and potentials.

It is potentially an ill-posed problem, depending on the assumptions, see e.g. [L. Borcea, Inv. Problems (2002)].

EIT problem is still an active research problem; it still poses challenging questions for mathematicians, numericians and experimentalists. This problem is discussed in detail e.g. in [36].

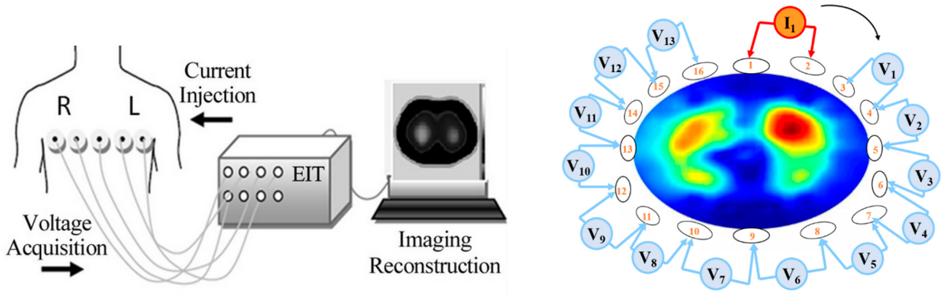


Figure 1.2: Electrical Impedance Tomography (EIT) for cardio-pulmonary monitoring: voltage measurements around the thorax using an EIT system with 16 electrodes. (R) Image extracted from C. Putensen et al., J. Clinical Medecine (2019).

Similar inverse problem in other real-world context Inverse problems based on similar diffusive equation arise in hydrology for example. The reader may consult e.g. [29, 36, 4] to read other standard instructive inverse problems in various technological contexts. Mathematical models are generally PDEs or integral equations.

Other complex examples and real-world are presented in next chapter, in particular the inverse problem in spatial hydrology which is analysed into detail throughout the manuscript.

1.3 General concepts

Basic formalism

In real-world problems, the measurements, denoted by z^{obs} ($z^{obs} \equiv y^{obs}$ if the observations are directly the model outputs), are *almost always incomplete, sparse or inaccurate.*

Moreover since the direct model represented by the operator $\mathcal{M}$ is un-perfect, measurements actually satisfies:

$$y^{obs} = \mathcal{M}(u) + \varepsilon \tag{1.2}$$

with ε a global error term incorporating both the observation errors and the structural model error: $\varepsilon = \varepsilon_{obs} + \varepsilon_{mod}$.
ε is defined as a stochastic field following an a-priori distribution, actually Gaussian when no other information is available.

For simplicity, we here assume that the observations are directly the model outputs: $z^{obs} = y^{obs}$.

1.3.1 Even the inversion of linear operators may not be trivial...

In the linear case, the direct model is represented by a matrix M.

Naïvely, solving the inverse problem as

$$u = M^{-1}\left(y^{obs} - \varepsilon\right) \tag{1.3}$$

may not work for few reasons.

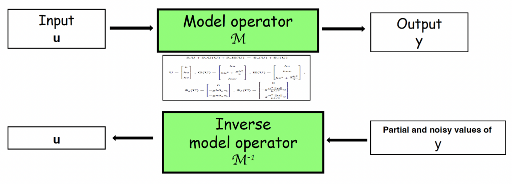
Two simple reasons are:
- observations y^{obs} can be not numerous enough therefore providing an undetermined problem,
- the error term ε can be unknown.

Moreover, M^{-1} can be ill-conditioned. This means that small variations of y implies large variations of u.
Note that M^{-1} well-conditioned (resp. ill-conditioned) implies that M well-conditioned (resp. ill-conditioned) too, since $\kappa(M) = \kappa(M^{-1})$.

"Mathematically invertible" does not mean "easily numerically invertible"...

In all the sequel, we define the *control-to-state operator* $\mathcal{M}$ (also called here the "model operator") as follows:

$$\boxed{\mathcal{M} : u \in \mathcal{U} \mapsto y \in \mathcal{Y}} \tag{1.4}$$



1.3.2 Well-posedness, ill-posedness

In the Hadamard sense[1], a model is well-posed if the following conditions hold:
i) it admits an unique solution,
ii) the solution depends continuously on the data or input parameters.

The first condition i) (existence and uniqueness) is related to the functional space the solution is sough in.

The second condition ii) is a stability condition. If this condition is not satisfied then any measurement or numerical error generates large errors in the model response i.e. a highly perturbed solution.
This condition may be re-read as follows: the control-to-state operator $\mathcal{M}$ is continuous.

Even if the direct problem is well-posed, in real-world problems the inverse problem is often ill-posed (or ill-conditionned as discussed later).
The stability refers to the sensitivity of the solution to small perturbations in the input data. A stable solution ensures that small changes in the data lead to small changes in the solution. Without stability, the solution may be physically meaningless or dominated by noise (e.g. measurement errors) or any uncontrolled input perturbation.

Note that thanks to the *open mapping theorem*, see e.g. [12]:
If $\mathcal{M}$ is linear and continuous with $\mathcal{U}$ and $\mathcal{Y}$ Banach spaces, then the inverse model operator $\mathcal{M}^{-1}$ is continuous.

Ill-posed inverse problems are somehow the opposite of well-posed direct problems.
Direct problems are usually the way that can be solved easily (compared to the inverse problem).

Actually, direct and inverse models are back-and-forth transmission of information between u and y.

In some way, if the direct model operator maps causes to effects, the inverse operator maps the effects to the causes.
In science and engineering, inverse problems often consist to determine properties of unmeasurable quantities (in space and/or in time).
The observations (data, measurements) are generally far to be complete or even accurate. Poor observations, both in quantity and quality, is one of the source of difficulties to solve inverse problems.

Exercise. Propose a PDE-based model which is well-posed in the Hadamard sense.

An answer: linear elliptic BVPs, coercitive in a Hilbert space V, may be well-posed in vertu of the Lax-Milgram theorem. □

1.3.3 Direct - inverse models: reverse frequencies

In real-world problems, the direct models generally represent the lowest frequencies of the modeled phenomena: $\min_i |\lambda(M)|$ is relatively large compared to noise frequencies, see e.g. Fig. 1.3.

The most energetic modes of the Fourier representation of a signal, here $y(u)$ the direct model output, are the lowest frequencies. Indeed, recall that: $\|\hat{y}(u)\|_2 =$...
Noises are high frequencies (therefore not energetic). Moreover, the spectrum of M and M^{-1} are related as: $\lambda_i(M^{-1}) = (\lambda_i(M))^{-1}$ for all i. Therefore, the highest frequencies of $y(u)$ are the lowest frequencies of $u(y)$.

As a consequence, separating noise from the inverse model output (e.g. in an image) is a difficult task.

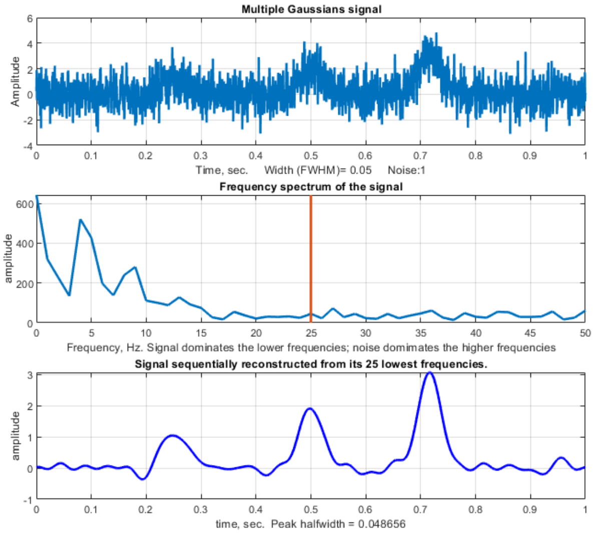


Figure 1.3: Direct models generally represent the lowest frequencies of the modeled phenomena (here a superimposition of Gaussians).

In real-life modeling, inverse problems are as common as direct problems. The inverse problem classes and techniques to solve them are extremely wide.
Inverse modeling is a fast growing topic, requiring both stochastic and deterministic skills.

The historical and common applications fields are in geosciences e.g. weather forecast, oil reservoirs, hydrology, neutronic (nuclear power plants), inverse scattering (seismology) and imaging (medical imaging, tomography, image restoration).

Data Assimilation (DA) methods aim at fusing data into mathematical models. DA may be viewed as a class of method aiming at solving some particular inverse problems.

Chapter 2

Basic tools

In this chapter, we revisit key classical algebraic tools that are essential for understanding and analyzing *finite-dimensional inverse problems, with a particular focus on regression problems involving data.* The outline is as follows.

Contents

2.1 Regression problems: least-squares solutions, SVD

Numerous excellent books may be consulted on the topic, e.g. [A. Bjorck, Numerical Methods for least-squares Problems, SIAM, Philadelphia, 1996].
Linear Regression is likely the most employed approximation technique, dating (at least) from I. Newton and J. Cassini in the 1700's.... Today, it remains widely used to fit parametrized curves to experimental data. Thus, it still may be qualified as the simplest form of the modern Machine Learning technique...

2.1.1 Linear least-squares problems

Let assume that we have m measurements $(z_i)_{1\leq i\leq m}$ we seek to describe by a "model".

To do so, we choose to consider the following *linear* model with n unknown parameters $(u_i)_{1\leq i\leq n}$:

$$\left\{ \begin{array}{rcl} a_{11}u_1 + ... + a_{1n}u_n & = & z_1 \\ ... & = & ... \\ a_{m1}u_1 + ... + a_{mn}u_n & = & z_m \end{array} \right.$$

We denote by $A = (a_{ij})_{1\leq i\leq m, 1\leq j\leq n}$, A a $m \times n$-matrix, by $z \in \mathbb{R}^m$ the observation vector and by $u \in \mathbb{R}^n$ the unknown input parameter.
This problem re-reads as:

$$\left\{ \begin{array}{l} \text{Given } A \in \mathbb{R}^{m\times n} \text{ and } z \in \mathbb{R}^m, \\ \text{find } u \in \mathbb{R}^n \text{ such that: } Au = z \end{array} \right. \tag{2.1}$$

The linear model A is here imposed. The basic linear least-squares problem problem may be perceived as an *identification parameter problem* given the data.

In the case there is as much parameters u_i as observations z_k (yes, that sounds weird...), i.e. $n = m$, the model is well-posed if and only if A is a full-rank matrix. In this case, it exists a unique set of parameters u describing exactly the data.

Still in the case $n = m$ but if the model is ill-posed in the sense rank$(A) < n$, then it exists solutions but they are not unique.
In this case, the kernel of A, $Ker(A)$, contains non zero vectors v such that: $Av = 0$. If u^* is a solution then $(u^* + v)$ with $v \in Ker(A)$ is also solution.
It is very unlikely to have $n = m$!...

In the case $n > m$ i.e. in the unlikely case there is more input parameters than observations, the system is under-determined. Generally, it exists solutions but they are non-unique.

In the case $n < m$ i.e. in the usual case there is less input parameters than observations, the system is over-determined. A-priori it does not exist any solution fitting all data.
Indeed, with m input parameters, it can exist a unique solution making fit the observations; but what about the extra $(n - m)$ "constraint equations" ?

Least square solution (in the linear case)

It is interesting to find a solution satisfying "at best" the system: a solution u minimizing the norm $\|Au - z\|$.

The Euclidian norm $\|.\|_2$ is associated to a *scalar product* on contrary to the norms $\|.\|_1$ and $\|.\|_\infty$.

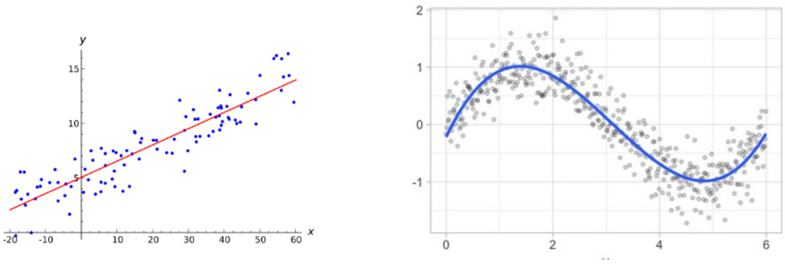


Figure 2.1: Linear least-squares problem (with here dense observations-measurements). (L) The simplest case: linear regression (two parameters to identify). (R) Degree 3 - polynomial regression.

Using the $\|.\|_2$ norm, the problem reads:

$$\boxed{\begin{cases} \text{Find } u^* \in \mathbb{R}^n \text{ such that:} \\ j(u^*) = \min_{\mathbb{R}^n} j(u) \text{ with } j(u) = \frac{1}{2}\|Au - z\|_{2,m}^2 \end{cases}} \tag{2.2}$$

It is an unconstrained optimization problem in a convex set.

The functional j reads: $j(u) = \dfrac{1}{2}(A^T Au, u) - (Au, z) + \dfrac{1}{2}(z, z)$. j is quadratic, convex since $A^T A$ is positive, hence j admits a minimum in $\mathbb{R}^n$.
Furthermore, if $A^T A$ is definite (it means $n \leq m$ and A is full rank, $\dim(\text{Im}(A) = m)$ then the solution is unique.
The gradient of j reads: $\nabla j(u) = A^T Au - A^T z$.

The linearity of the model A (A is a matrix) implies that the optimal solution u^* satisfies the following *optimality condition*:

$$\boxed{A^T Au = A^T z} \tag{2.3}$$

This is the *normal equations*.

Examples

The reader can find numerous well documented examples with corresponding Python codes available on the web, e.g. on the https://towardsdatascience.com web site [1].

A very basic example Below is presented a very basic simple. Data in (x,y) are generated (synthetic dataset). The matrix A is assembled by stacking x and a column of ones. The least-squares solution is computed (linear algebra). The data points along with the fitted curve obtained from the least-squares solution are plotted.

[1] https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570

```
# Basic example of least-squares solution approximating a set of scalar values
import numpy as np
import matplotlib.pyplot as plt

# Generate some synthetic data: linear generation ! To be complexify...
x = np.linspace(0, 1, 101)
y = 1 + x + x * np.random.random(len(x))

# Assemble matrix A
A = np.vstack([x, np.ones(len(x))]).T

# Perform least-squares regression
alpha = np.dot(np.linalg.inv(np.dot(A.T, A)), np.dot(A.T, y[:, np.newaxis]))

# Plot the data points along with the least-squares regression
plt.plot(x, y, 'ro', label='Data')
plt.plot(x, np.dot(A, alpha), 'k--', label='Fit')
plt.xlabel('x'); plt.ylabel('y'); plt.title('least-squares regression')
plt.legend(loc='upper left')
plt.show()
```

Full rank case* *This paragraph is a "not compulsory" one.*
In the case A full rank (i.e. $\dim(\mathrm{Im}(A)) = m$), then the solution is unique since A^TA is symmetric positive definite. Even if A sparse, then A^TA is a-priori non sparse; also $u_2(A^TA) = (u_2(A))^2$, hence the normal equations can be an ill-conditioned system. Then, a good algorithm to solve the normal equations is a-priori not the Cholesky algorithm, especially if m large. Remembering that the 2-norm is preserved under orthogonal transformations, a better option is to solve the following equivalent system:

$$\min_{u \in \mathbb{R}^n} \frac{1}{2} \|QAu - Qz\|_{2,m}^2$$

with Q an orthogonal matrix.
By performing QR-factorizations (Householder's method), the resulting linear system to be inverted is a triangular $n \times n$ system, with its original conditioning number $K_2(A)$ preserved.

2.1.2 Singular Value Decomposition (SVD) for linear inverse problems

This section is a "not compulsory" section
The Singular Value Decomposition (SVD) is a widely employed and powerful technique used in linear algebra and data analysis. It can be applied to solve linear inverse problems, such as image reconstruction, noise reduction, system identification etc. SVD is an important tool to analyze *linear* inverse problems (the model A is a matrix).
It is the central tool to build up reduced models by the Proper Orthogonal Decomposition (POD) method and computing PCA.

Let us recall what is the Singular Value Decomposition (SVD) of a matrix A.

Recalls on the SVD

Let A be a $m \times n$-rectangular matrix.

In a nutshell, the singular values of A, denoted by σ_i, are the squared roots of the eigenvalues of AA^T (or A^TA). They allow the decomposition of A into a basis formed by orthogonal eigenvectors.
More precisely, given A $m \times n$-rectangular matrix, of rank r, $r < m$, the SVD of A reads as follows, see e.g.[]:

$$A = V\Sigma u^T = \sum_{i=1}^{r} \sigma_i v_i u_i^T$$

where:

. Σ is the $r \times r$-diagonal matrix containing the singular values σ_i of A:

$$(\sigma_i)^2 = \lambda_i(A^TA) = \lambda_i(AA^T) \text{ for all } \lambda_i \neq 0,\ 1 \leq i \leq r$$

$$0 < \sigma_r \leq ... \leq \sigma_1$$

. $V = (V_1, ..., V_r)$, $m \times r$-matrix, contains the unit orthogonal eigenvector basis of AA^T: $(AA^T)V = V\Sigma^2$ with $V^TV = \mathcal{I}_r$,

. $W = (W_1, ..., W_r)$, $n \times r$-matrix, contains the unit orthogonal eigenvector basis of A^TA: $(A^TA)W = W\Sigma^2$ with $u^TW = \mathcal{I}_r$.

The vectors of V constitute an unit orthogonal basis of $Im(A) \subset \mathbb{R}^m$, while the vectors of W constitute an unit orthogonal basis of $Ker^T(A) \subset \mathbb{R}^n$.

From the SVD, a *pseudo-inverse* (also called generalized inverse) of the rectangular matrix A can be defined as follows:

$$A^{-1} = W\Sigma^{-1}V^T$$

The SVD provides optimal low-rank approximations to matrices. Indeed, we have:

Theorem 2.1. *(Eckart-Young theorem). The $r-$rank SVD truncation of a $m \times n$-matrix A is the optimal rank approximation to A in the least- squares sense.*
Indeed, the matrix $A_k = \sum_{i=1}^{n} \sigma_i v_i w_i^T$ with $k \leq r$ is optimal in the following sense:

$$\|A - A_k\|_F = \min_{B \in \mathbb{R}^{m \times n},\ rank(B) \leq k} \|A - B\|_F = \left(\sum_{m=(k+1)}^{r} \sigma_m^2 \right)^{1/2}$$

where $\| \cdot \|_F$ denotes the Frobenius norm.

Recall: $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\sum_{i=1}^{r} \sum_{j=1}^{n} \sigma_i^2} = \sqrt{trace(A^TA)}$ (or trace(AA^T)).

Solving the linear least-squares problem using SVD

Let us go back to the linear least-squares problem. In (2.2), the residual to be minimized satisfies:

$$\|Au - z\|_{2,m}^2 = \|V^T(V\Sigma u^T u - z)\|_{2,m}^2 = \|\Sigma u^T u - V^T z\|_{2,m}^2$$

since V is an orthogonal matrix.
Therefore the least-squares solution u^* satisfies:

$$\boxed{u^* = A^{-1}z = W\Sigma^{-1}V^T z = \sum_{i=1}^{r} \left(\frac{1}{\sigma_i}\right) v_i^T w_i \; z} \tag{2.4}$$

with the singular values σ_i ordered in decreasing order (decreasing gradually to 0).

A few consequences of this explicit expression of the optimal solution u^*.

- Smaller singular values are, greater they amplify the errors contained in the data z. Vanishing singular values produce instabilities in the computation of the expression above.
- The more the number of singular values are taken into account (from 1 to r at maximum), the more the data errors are amplified.
- This SVD-based expression provides a hierarchical approximation to the inverse problem solution u^* (in the eigenvectors coordinate system).

A few other remarks

- From this expression, one may want to "regularize" the solution u^* of an ill-conditioned or ill-posed linear inverse problem by truncating the sum at a given order r_0 strictly lower than r.
- However, r_0 has to be not too large, not too small... No universal criteria exist to quantify a good truncation rank r_0... This has to be empirically determined from expertise and potentially on error criteria such as $\|u_{r_0} - u^*\|/\|u^*\|$ in a given norm (typically the 2-norm or the ∞-norm).

- Note that computing the SVD of large matrices is computationally expensive: the complexity is
- Again, the SVD may be a useful tool to analyze inverse problems which are linear only. (The model A is here a matrix).

Examples

The reader can find numerous well documented examples with corresponding Python codes available on the web, e.g. on the https://towardsdatascience.com web site [2].

[2] hhttps://towardsdatascience.com/understanding-singular-value-decomposition-and-its-application-in-data-science-388a54be95d

A basic example Below is presented a very basic simple. A noisy signal (consisting of a sinusoidal wave with added Gaussian noise) is first generated. We then perform a SVD on the signal. By selecting a subset of singular values (the first k singular values), we can reconstruct the signal. The original and reconstructed signals are plotted.

```
# SVD to reconstruct a 1D signal
import numpy as np
import matplotlib.pyplot as plt

# Generate a noisy signal
t = np.linspace(0, 1, 100)
signal = np.sin(2 * np.pi * 5 * t) + np.random.normal(0, 0.1, 100)

# Perform SVD
U, S, V = np.linalg.svd(signal)

# Reconstruct the signal using a subset of singular values
k = 10
reconstructed_signal = U[:, :k] @ np.diag(S[:k]) @ V[:k, :]

# Plot the original and reconstructed signals
plt.figure(figsize=(8, 4))
plt.plot(t, signal, label='Original Signal')
plt.plot(t, reconstructed_signal, label='Reconstructed Signal')
plt.xlabel('Time'); plt.ylabel('Amplitude'); plt.title('SVD Applied to an Inverse Problem')
plt.legend()
plt.show()
```

2.1.3 Non linear least-squares problems

Let us consider the same problem as previously: set up a model describing "at best" m available data z_i, $1 \leq i \leq m$ but based on a *non linear* model $\mathcal{M}$. In this case, the corresponding least-squares formulation reads:

$$\boxed{j(u^*) = \min_{U \subset \mathbb{R}^n} j(u) \text{ with } j(u) = \frac{1}{2} \|\mathcal{M}(u)\|_{2,m}^2} \tag{2.5}$$

with $\mathcal{M}$ defined from $\mathbb{R}^n$ into $\mathbb{R}^m$, $\mathcal{M}(\cdot)$ *non linear.*

Again, it is an unconstrained optimization problem in a convex set.

In the previous linear case, $M \equiv \mathcal{M}$ was equal to: $M(u) = Au - z$.

Note that neither the existence of a solution, nor its uniqueness is guaranteed without assumptions on the non linear operator $M(\cdot)$ and/or the space solutions U.In a nutshell, non linear problems are much more difficult than linear ones. Below, and more generally in this course, one consider heuristic computational algorithms only.

Examples

Below a very few examples of non linear least-squares problems. Many other examples can be found e.g. in [15] or e.g. on the https://towardsdatascience.com web site.

Example: standard fitting problem The goal is to fit at best a (dense) dataset; the regression problem plotted in Fig. 2.2(L).
The chosen functional (”model”) is: $j(u_1, \cdots, u_4) = u_1 \exp(u_2 x) \cos(u_3 x + u_4)$ with the four parameters $u_i,\ i = 1, \ldots, 4$, to identify.

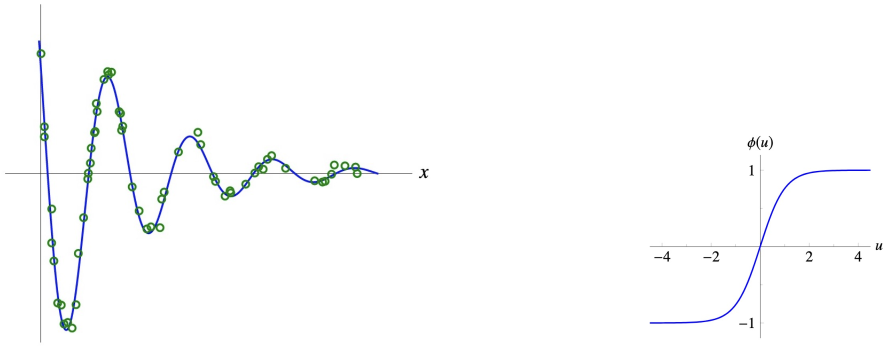


Figure 2.2: non linear least square problems: examples. (L) A fitting problem (with dense data); 4 parameters to identify. (R) The (differentiable) sigmoid function employed to solve binary classification problems.

Example: classification problem A basic way to solve a binary classification problem is to solve the non linear least square problem defined by:

$$j(u_1, \cdots, u_p) = \sum_{i=1}^{m} \left(\Phi(u_1 f_1(x_i) + \cdots + u_p f_p(x_i)) - d_i\right)^2$$

where the functions $f_j(\cdot)$ are given.
$\Phi(u)$ is he sigmoidal function; $\Phi(u) = \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)}$. Φ is a differentiable function approximating the sign function, Fig. 2.2(R).

Example: The location problem by Global Navigation System Satellites (GPS, Galileo etc). The location problem by Global Navigation System Satellites (GNSS) consists to estimate the location value x, $x \in \mathbb{R}^3$, given measurements.
The measurements are distances d_i^{obs} from m given locations a_i, $i = 1, \cdots, m$ (the m satellites):

$$d_i^{obs} = \|a_i - x^{exact}\|_{2,\mathbb{R}^3} + \varepsilon_i \quad i = 1, .., m$$

where ε denotes the inacurracy of the measurements (the ”noise”).

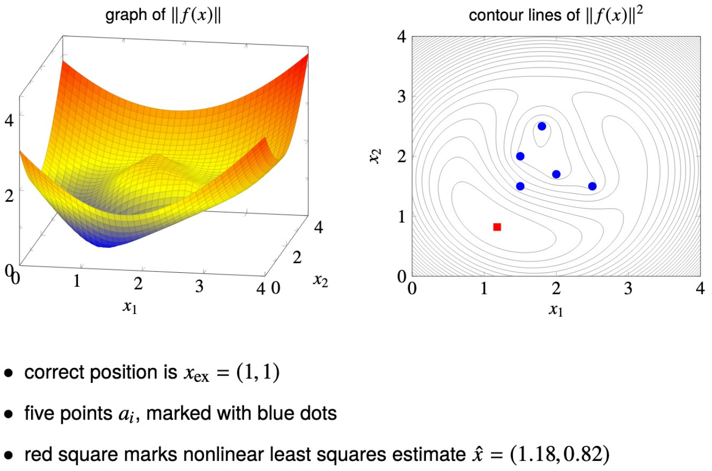
We define the functional $j(\cdot)$, $j : \mathbb{R}^3 \rightarrow \mathbb{R}$, as:

$$j(u) = \sum_{i=1}^{m} \left(\|a_i - u\|_{2,\mathbb{R}^3} - d_i^{obs} \right)^2$$

The location problem consists to solve the following problem:

$$\text{Find } x^* = arg \min_{x \in \mathbb{R}^3} j(x) \qquad (2.6)$$

Note that $j(\cdot)$ is a-priori not convex!... See Fig. 2.3.



- correct position is $x_{\text{ex}} = (1,1)$
- five points a_i, marked with blue dots
- red square marks nonlinear least squares estimate $\hat{x} = (1.18, 0.82)$

Figure 2.3: The location problem by GNNS is a non convex non linear least square problem... (Image extracted from Vandenberghe's UCLA course).
(L) Graph of $j(u)$: non convex. (R) Iso-values of $j(u)$, the ref points (blue) and an approximate solution (red square).

Another example of approximation with the corresponding Python code Below is presented a basic example. We consider a nonlinear model function model(x, params). The data are generated as the "true" model output plus some Gaussian noise. We then define an objective function objective(params) that computes the difference between the observed data and the model predictions for a given set of parameters. Finally, the nonlinear least-squares fit is obtained by using the least_squares function from SciPy. The solutions are plotted.

```
# A non linear square approximation problem
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

# Define the function to be fitted
def model(x, params):
    return params[0] * np.exp(params[1] * x)

# Generate some noisy data
x = np.linspace(0, 10, 100)
true_params = [0.1, 0.3]
y_true = model(x, true_params)
y_noisy = y_true + np.random.normal(0, 0.1, len(x))

# Define the objective function for least-squares
def objective(params):
    return y_noisy - model(x, params)

# Perform the nonlinear least-squares fit
initial_params = [0.2, 0.2]
result = least_squares(objective, initial_params)

# Extract the fitted parameters
fitted_params = result.x


# Generate the fitted curve
y_fitted = model(x, fitted_params)

# Plot the original data and the fitted curve
plt.figure(figsize=(8, 4))
plt.plot(x, y_noisy, 'bo', label='Noisy Data')
plt.plot(x, y_true, 'g-', label='True Curve')
plt.plot(x, y_fitted, 'r-', label='Fitted Curve')
plt.xlabel('x'); plt.ylabel('y'); plt.title('Nonlinear least-squares Fit')
plt.legend()
plt.show()
```

Optimality condition(s) & derivatives expressions

We have: $j(u) = \frac{1}{2}\|\mathcal{M}(u)\|^2_{2,m}$ with $\mathcal{M}(u) = \big(\|a-u\|_{2,\mathbb{R}^3} - d^{obs}\big)$.
The necessary first order optimality condition for the general non linear square problem (2.5) reads:

$$\boxed{\nabla j(u) = 0} \tag{2.7}$$

The Jacobian of $\mathcal{M}$ reads: $D\mathcal{M}(u) = \Big(\frac{\partial \mathcal{M}_i}{\partial u_j}(u)\Big)_{1\leq i\leq m, 1\leq j\leq n}$.
The Hessian for each component $\mathcal{M}_i(u), 1 \leq i \leq m$ reads: $D^2\mathcal{M}_i(u) = \Big(\frac{\partial^2 \mathcal{M}_i}{\partial u_l \partial u_j}(u)\Big)_{1\leq l\leq m, 1\leq j\leq n}$.

Then, the gradient and the Hessian of $j(u)$ read:

$$\nabla j(u) = D\mathcal{M}^T(u)\mathcal{M}(u)$$

$$H_j(u) \equiv D^2 j(u) = D\mathcal{M}^T(u)D\mathcal{M}(u) + \sum_{i=1}^{m} \mathcal{M}_i(u) D^2\mathcal{M}_i(u)$$

Expressions in the linear case In this case we have: $u \mapsto \mathcal{M}(u)$ linear.
Then, the gradient reads: $\nabla j(u) = A^T\mathcal{M}(u) = A^T(Au - z)$ and the Hessian reads: $H_j(u) = A^TA$ since the term $D^2\mathcal{M}_i$ in the Hessian expression vanishes.

Exercise 2.2. *Verify the expression of the gradient and the Hessian above.*

Resolution algorithms: Newton, Gauss-Newton and Levenberg-Marquardt

The **Newton-Raphson algorithm** applied to the optimality condition $\nabla j(u) = 0$ consists to solve at each iteration:

$$\boxed{H_j(u^n) \cdot \delta u = -\nabla j(u^n) \ ; \quad u^{n+1} = u^n + \delta u}$$

Newton-Raphson's algorithm requires the computation and the inversion of the Hessian of j. For complex real-world models, the computation of H_j is often prohibitive because too complex or too CPU-time consuming.

Exercise 2.3. *Verify that the Newton algorithm applied to the equation* $(\nabla j(u) = 0)$ *reads as above.*

The principle of the **Gauss-Newton algorithm** is to consider in the Newton-Raphson algorithm an approximation of the Hessian by omitting the second order term:

$$\tilde{H}_j(u) \equiv D\mathcal{M}^T(u)D\mathcal{M}(u) \approx H_j(u) \tag{2.8}$$

This provides the following linear system to solve at each iteration:

$$\boxed{\tilde{H}_j(u) \cdot \delta u = -\nabla j(u^n) \ ; \quad u^{n+1} = u^n + \delta u} \tag{2.9}$$

with (recall) $\nabla j(u^n) = D\mathcal{M}^T(u^n)\mathcal{M}(u^n)$.

Note that the linear system to be inverted is symmetric positive, and definite if $D\mathcal{M}(u^n)$ is full rank.
The Gauss-Newton method is observed to be very efficient if $DM^T(u^n)$ is full rank and if $M(u)$ small when close to the solution. On the contrary it becomes inefficient if these two conditions are not satisfied and/or if the model is locally "far to be linear" i.e. if the term $\sum_{i=1}^{m} M_i(u)D^2M_i(u)$ is non negligible (or even worse, dominant).

Finally, a good alternative to solve non linear least square problems is the **Levenberg-Marquardt algorithm**, see e.g. [4].

The Levenberg-Marquardt algorithm can be considered a damped version of the Gauss-Newton algorithm.
It combines aspects of a gradient descent algorithm with the Gauss-Newton algorithm described above.

2.2 Ill-posed and ill-conditionned inverse problems: regularization

2.2.1 Generalities

Regularization is a mathematical framework designed to address issues in solving ill-posed or ill-conditioned problems.The goal of regularization is to reformulate the problem so that the solution becomes stable, well-defined, and robust to perturbations in the data. The general framework of regularization can be understood through the lens of stability, existence, and uniqueness of solutions.

In the *simple least-squares problem* described earlier, see (2.1), if the matrix A is full rank $(\dim(\mathrm{Im}(A)) = m)$, the least-squares solution u^* which satisfies the normal equations (2.3), is unique. In this case the inverse problem is well-posed.

In the *simple linear least-squares problem*, even if A is full rank, the system may still be ill-conditioned. This occurs when the smallest singular value of A is close to zero, making the problem sensitive to noise or perturbations in the data[3].

Such ill-conditioning is a common challenge in real-world problems. To address this issue and improve the stability of the solution to the inverse problem, it becomes necessary to introduce "regularization".
Regularization involves selecting a solution with desirable properties by imposing additional constraints that encourage specific characteristics such as "smoothness". From a mathematical standpoint, this approach restricts the solution to a smaller, more structured functional space, ensuring it adheres to predefined regularity conditions while still approximating the observed data.

From a mathematical perspective, this means seeking the solution in a smaller, more restrictive functional space.

For instance, given a bounded domain Ω, the space of continuously differentiable functions $C^1(\Omega)$ is strictly contained within the space of bounded functions $L^\infty(\Omega)$. Naturally, functions in $C^1(\Omega)$ exhibit greater smoothness (or regularity) than those in $L^\infty(\Omega)$.
In practical terms, regularization often takes the form of an augmented cost function, as the

[3] Recall that the condition number of a matrix A is given by $\kappa(A) = \frac{\max_i \sigma_i(A)}{\min_i \sigma_i(A)}$. For a normal matrix A, this ratio equals $\frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}$, where $\lambda_i(A)$ are the eigenvalues of A.

Tikhonov-like regularization does (see in next section) .

On the use of the SVD for regularizing the solution u^* In the case of a *linear* inverse problem, we have previously shown (see Section 2.1.2) that the SVD may be good a useful tool to regularize the optimal solution u^*.
However: a) a good truncation rank r_0 may be difficult to determine; b) the SVD decomposition does not apply to non linear problems; c) the computation of the SVD is hardly tractable in large dimension cases.

2.2.2 Tikhonov's regularization

A. Tikhonov, Russian mathematician, 1906-1993). Note this method may be due to Phillips and Miller too.
The most classical method to regularize large dimensions optimization problems ($u \in \mathbb{R}^m$ with m large) consists to add the so-called Tikhonov's regularization term or variants.
The so-called Tikhonov's regularization aims at augmenting the minimized cost function $j(u)$.

In the linear least-squares problem case In this case, the Tykhonov regularization consists to search $u^* = \arg\min_{U \subset \mathbb{R}^n} j_\alpha(u)$ with:

$$\boxed{j_\alpha(u) = \frac{1}{2}\|Au - z\|^2_{2,m} + \alpha_{reg}\frac{1}{2}\|Cu\|^2_{2,n}} \tag{2.10}$$

where α_{reg} is a positive scalar setting the balance between the "model fidelity" term and the "regularity term".
Above, C denotes a linear operator (an invertible matrix).

The simplest choice for C is $C = Id$. In this case, the corresponding least-squares solution is the one with the minimal 2-norm.

The normal equations corresponding to the minimization problem above read:

$$\boxed{(A^T A + \alpha_{reg}^2 C^T C)\ u = A^T z}$$

With $C = Id$, the regularization acts as a diagonal augmentation.

Independently on the case We seek to add regularization terms that are differentiable, strictly convex.

Of course, the obtained solution u^* depends on the regularization parameter α_{reg}.

By including the Tikhonov regularization term in the objective function, the optimization algorithm seeks to find a balance between minimizing the model fidelity and the magnitude of the parameter u in a given norm (assuming that $C^{1/2}$ defines a norm).

The parameter α_{reg} determines the balance between stability and "fidelity": larger α_{reg} increases stability but may introduce bias, while smaller α_{reg} reduces bias but may amplify instability.

In summary, the expected effects of a regularization term are:

- An improved stability: the introduced regularization should mitigates the effects of noise or ill-conditioning by amplifying the role of (chosen) well-behaved components of the solution.
- A reduced "ambiguity": by incorporating *prior knowledge*, e.g., smoothness, sparsity, or physical constraints, regularization narrows down the set of possible solutions to those consistent with the expected ones.
- A more robust numerical optimization: regularization transforms the inverse problem into a well-posed optimization problem, thats should ensure the uniqueness and stability of the solution under small perturbations in the data. (The existence of solutions is here implicitly assumed).

ToDo: REFAIRE FIGS

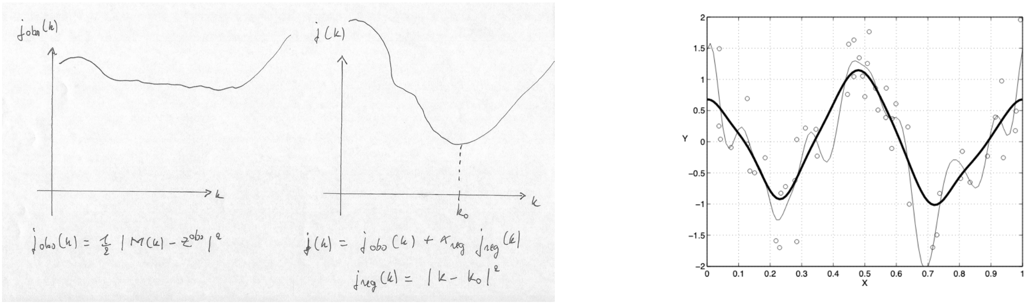


Figure 2.4: Tikhonov's regularization. (L) A typical "poorly convex" functional $j_{data}(\cdot)$: ill-conditioned minimisation problem. (M) Regularized functional to be minimized: $j(.) = j_{data}(.) + \alpha_{reg} j_{reg}(.)$ with $j_{reg}(u) = \|u - u_0\|$, u_0 a *prior* value. (R). Data fitting (in the LS sense) without and with regularization (Image extracted from a MIT course).

In summary, regularization is a critical tool for solving ill-posed or ill-conditioned problems, bridging the gap between theoretical formulations and practical, robust solutions. Regularization is a wide and difficult topic. The reader may consult excellent books such as e.g. [18, 26, 23, 29, 4] which adresses this topic in different manners.

Example (with the corresponding Python code)

Below is presented a basic example where a nonlinear model function model(x, params) is defined. A regularization function regularization(params, alpha) that computes the Tikhonov regularization term given a set of parameters and a regularization parameter alpha is also defined. We generate some noisy data by adding Gaussian noise to the true model output. We then define an objective function objective(params) that computes the difference between the observed data and the model predictions for a given set of parameters. Finally, we use the least_squares function from SciPy to perform the nonlinear least-squares fit with Tikhonov regularization by appending the regularization term to the objective function. The different fields are plotted.

```
# Minimization by Levenberg-Marquardt of a functional enriched with a Tikhonov regularization term.
# The weight parameter may be better be tuned...
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

# Define the function to be fitted
def model(x, params):
    return params[0] * np.exp(params[1] * x)

# Define the Tikhonov regularization term
def regularization(params, alpha):
    return alpha * np.sum(params ** 2)

# Generate some noisy data
x = np.linspace(0, 10, 100)
true_params = [0.1, 0.3]
y_true = model(x, true_params)
y_noisy = y_true + np.random.normal(0, 0.1, len(x))

# Define the objective function for least-squares with regularization
def objective(params):
    return y_noisy - model(x, params)

# Perform the nonlinear least-squares fit with Tikhonov regularization
initial_params = [0.2, 0.2]
alpha = 0.01  # Tikhonov regularization parameter
result = least_squares(lambda params: np.append(objective(params), regularization(params, alpha)), initial_

# Extract the fitted parameters
fitted_params = result.x

# Generate the fitted curve
y_fitted = model(x, fitted_params)

# Plot the original data and the fitted curve
plt.figure(figsize=(8, 4))
plt.plot(x, y_noisy, 'bo', label='Noisy Data')
plt.plot(x, y_true, 'g-', label='True Curve')
plt.plot(x, y_fitted, 'r-', label='Fitted Curve')
plt.xlabel('x'); plt.ylabel('y'); plt.title('Nonlinear least-squares Fit with Tikhonov Regularization')
plt.legend()
plt.show()
```

A few remarks

- The Tikhonov's regularization term can be easily added even in large-scale optimization problems.

- In real-world modeling problems, the regularization operator C is often defined as a differential operator (first or second order) or as a Fourier operator aiming at filtering the highest frequencies.
Indeed, as already discussed numerous real-world phenomena (therefore the models) have the effect of low-pass filtering. As a consequence, the inverse operator acts as a high-pass filter....
The eigenvalues (or singular values) are the largest in the reverse mapping when they are the smallest in the forward mapping.
Amplifying high frequency is not a desirable feature since it amplifies noise and uncertainties e.g. noise in data measurements.

- In statistics, the Tikhonov regularization is called the *ridge regression* method.

- In Machine Learning (ML), this corresponds to the so-called *weight decay* method.

- Other regularization terms can be considered in particular in L^q-norm particurlarly with $q = 1$.

This case introduces compress sensing effects on the parameter u due to the 1-norm property, see e.g.[17].
This corresponds to the so-called *LASSO* problem. Obviously, this is not a least-square problem anymore.
The use of 1-norm provides a convex regularization but not differentiable therefore leading to non-differential convex optimization problems.

Setting the regularization parameter α_{reg}: a critical choice

The regularization parameter α_{reg} tunes the balance between the misfit to data (the fidelity to the model) and the smoothness of the solution. The solution of the inverse problem depends on this parameter; however, its "best" value is a-priori unknown.
Setting the value of α_{reg} is a crucial step of the inverse problem solving.
The parameter α_{reg} is critical and can be chosen using (generalized) cross-validation, the basic discrepancy principles (also called Morozov's principle) or the L-curve method.
For details the reader may consult e.g. [23] Chapter 5 or [P. C. Hansen, Ranu-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion, SIAM, Philadelphia, 1998].

The L-curve concept is briefly described in next paragraph.

2.2.3 L-curve for bi-objective optimization*

This is a "to go further section"

Let us consider the minimization problem $u^* = arg \min_u j(u)$ with:

$$j(u) = j_{obs}(u) + \alpha_{reg} j_{reg}(u)$$

The introduction of the regularization term $j_{reg}(u)$ leads to a bi-objective optimization.
As already mentioned, the weight parameter value α_{reg} has to be a-priori set. Obviously the computed optimal solution depends on α_{reg}.
No magic criteria nor method exist to determine an optimal value of α_{reg}.

The weight coefficient α_{reg} has to respect a "good" trade-off between the two terms $j_{misfit}(\cdot)$ and $j_{reg}(\cdot)$.

In the case of large dimensional problems, an expertise of the modeled phenomena likely remains a good manner to determine an optimal value of α_{reg}... The so-called L-curve enables to visualize this trade-off.

Concept of L-curve

The L-curve is a concept in bi-objective optimization that helps visualize the trade-off between two conflicting objectives, see Fig. 2.6.
It is a graphical representation of the Pareto front (which is, recall, the set of all non-dominated solutions in a multi-objective optimization problem).

In bi-objective optimization, the goal is to find a set of solutions that are not dominated by any other better solutions. The L-curve shows the relationship between the two objectives and helps the modeler understand the trade-offs involved in satisfying the two objectives...

Example of L-curve (with the corresponding Python code)

Let us consider the following basic optimization problem in $\mathbb{R}^2$:

$$\min_x f_1(x) = 100(x_1^2 + x_2^2) \text{ and } \max_x f_2(x) = -(x_1 - 1)^2 - x_2^2$$

such that:

$$g_1(x) = 2(x1 - 0.1)(x1 - 0.9) \leq 0 \ , \quad g_2(x) = 20(x1 - 0.4)(x1 - 0.6) \geq 0$$

$$-2 \leq x_1 \leq 2 \ , \quad -2 \leq x_2 \leq 2$$

This optimization problem has two objectives functions which are subject to two inequality constraints. The goal is to find the set of solutions that optimize both objectives while satisfying the constraints. To visualize the trade-off between the objectives, we can try to analyse

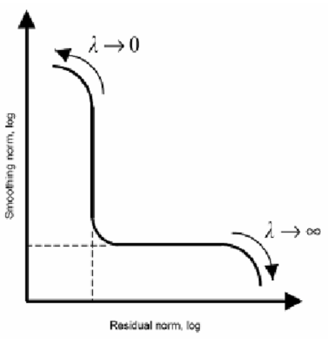


Figure 2.5: Bi-objective optimization: a typical L-curve (here in a clear pattern...). The optimal computed optimal solution depends on the weight parameter α_{reg} (here denoted by λ). The L-curve: values of the two objective functions (misfit and regularization terms) for different values of the weight parameter $\alpha_{reg} \equiv \lambda$.

plot the L-curve.
This is what the Python code below does (using the pymoo library).

```
# Bi-objective optimization problem: analyse of the L-curve
import numpy as np
from pymoo.model.problem import Problem
from pymoo.algorithms.nsga3 import NSGA3
from pymoo.factory import get_sampling, get_crossover, get_mutation
from pymoo.optimize import minimize

class MyProblem(Problem):
    def __init__(self):
        super().__init__(n_var=2, n_obj=2, n_constr=0, xl=np.array([-2, -2]), xu=np.array([2, 2]))

    def _evaluate(self, x, out, *args, **kwargs):
        f1 = 100 * (x[:, 0] ** 2 + x[:, 1] ** 2)
        f2 = -(x[:, 0] - 1) ** 2 - x[:, 1] ** 2
        out["F"] = np.column_stack([f1, f2])

problem = MyProblem()
algorithm = NSGA3(
    pop_size=100,
    sampling=get_sampling("real_random"),
    crossover=get_crossover("real_sbx", prob=0.9, eta=15),
    mutation=get_mutation("real_pm", eta=20),
)
res = minimize(problem, algorithm)

# Plotting the L-curve
import matplotlib.pyplot as plt

plt.scatter(res.F[:, 0], res.F[:, 1])
plt.xlabel("Objective 1"); plt.ylabel("Objective 2"); plt.title("L-curve")
plt.show()
```

2.2.4 Adaptative regularization & Morozov's principle*

This is a "to go further section"

Firstly let us note that the truncation order in the Truncated SVD, see (2.4), plays a similar role to the regularization parameter α_{reg} in Tikhonov's regularization, see e.g. [23] Chapter 4

for further discussions. In particular , the regularization parameter α_{reg} tending to 0 plays a similar role to the truncation rank tending to the maximal value.
Moreover, Tikhonov's regularization is convergent in the sense that u_{reg} converges to solution u^* when α_{reg} tending to 0 under certain conditions, see e.g. [18] Chapter 10 for details and proofs.

Adaptive regularization strategy is an approach used in numerical optimization to improve the performance of optimization algorithms. It involves adjusting the regularization parameter during the optimization process to achieve better results. The success of this regularization approach heavily depends on the Tikhonov parameter tuning strategy but also on the dimension of the projection subspace.

If defining the weight parameter $\alpha > 0$ of the regularization term[4] is set as a decreasing sequence as $\alpha^{(0)} > \cdots > \alpha^{(n)} > 0$, $n = 1, \cdots, n^*$, then the optimization procedure may be faster and more accurate, see e.g. [18, 26] and references therein.

Let us consider the unperturbed ("perfect") estimation problem: $z_{obs} = M(u)$.
Given a noise level ε, u^ε denotes the solution of:

$$z^\varepsilon_{obs} = M(u^\varepsilon) + \varepsilon \tag{2.11}$$

The stop iteration number $n^*(\varepsilon)$ may be chosen through the *Morozov' discrepancy principle* which reads:

$$\boxed{\tau_1\varepsilon \leq \|z^\varepsilon_{obs} - M(u^\varepsilon_{n^*})\| \leq \tau_2\varepsilon} \tag{2.12}$$

with $1 \leq \tau_1 < \tau_2$.

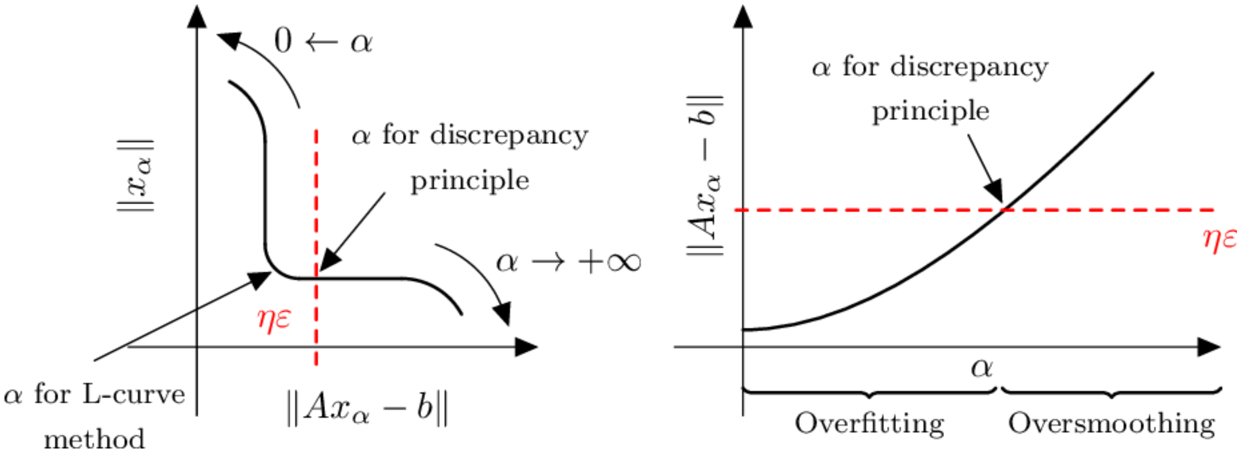


Figure 2.6: Bi-objective optimization and functional values obtained with different weight parameter values α_{reg}: (L) the L-curve criteria; (R): The Morozov's principle.

[4] The subscript $_{reg}$ is here skipped for sake of clarity

The weight parameter sequence $\alpha^{(n)}$ may be defined for example as, see e.g. [26] Chapter 4 for further discussions:

$$\alpha^{(n)} = \alpha^{(0)} q^{[n/n_0]}, \quad n = 1, \cdots, n^* \tag{2.13}$$

where $n_0 > 1$ is the number of iteration for each $\alpha^{(n)}$, $[m]$ returns the maximum integer smaller than m, $\alpha^{(0)}$ and q are given constants, $\alpha^{(0)} > 0$, $0 < q < 1$.
The values of $\alpha^{(0)}$, q, n_0 are chosen experimentally e.g. as: $q = 0.5$, $n_0 = 5$ and $\alpha_0 = 1$.

The stop iteration n^* may be then chosen according to (2.12).

Example of application of the Morozov's principle (with the corresponding Python code)

Below a code example which apply the Morozov's principle to a simple inverse problem with noisy data. It iteratively adjusts the regularization parameter α until the discrepancy between the forward model $M(u)$ and the noisy data z_{noise} falls within the desired bounds defined by τ_1 and τ_2.

```
# Application of the Morozov's principle
import numpy as np
import matplotlib.pyplot as plt

# Define the data and the operator
x_true = np.linspace(0, 10, 100)
u_true = np.sin(x_true)
M = lambda x: np.sin(x)

# Add noise to the data
noise_level = 0.1
z_noisy = u_true + noise_level * np.random.randn(len(u_true))

# Define the Tikhonov functional and the penalty term
alpha = 0.1  # Initial guess for alpha
q = 2  # Power for the norm in the functional
Psi = lambda x: np.linalg.norm(x, ord=q)**q

# Apply the Morozov's discrepancy principle
tau1 = 0.5  # Lower bound for the discrepancy
tau2 = 2.0  # Upper bound for the discrepancy

while True:
    # Solve the regularized problem
    x_alpha = np.linalg.solve(M(x_true).T @ F(x_true) + alpha * Psi(x_true), F(x_true).T @ z_noisy)

    # Compute the discrepancy
    discrepancy = np.linalg.norm(M(x_alpha) - z_noisy)

    if tau1 * noise_level <= discrepancy <= tau2 * noise_level:
        break

    # Update alpha using a bisection method
    if discrepancy < tau1 * noise_level:
        alpha *= 0.5
    else:
        alpha *= 2.0

# Plot the results
plt.plot(x_true, z_noisy, 'bo', label='Noisy data')
plt.plot(x_true, M(x_alpha), 'r-', label='Regularized solution')
plt.plot(x_true, z_true, 'k--', label='True solution')
plt.legend(); plt.xlabel('x'); plt.ylabel('u'); plt.title('Morozov\'s Discrepancy Principle')
plt.show()
```

Chapter 3

Real-world examples of inverse problems

Please consult the supplementary material

Part II

Data Assimilation (DA): Sketch of Methods

Chapter 4

DA in a nutshell

The outline of this chapter is as follows.

Contents

4.1 Data Assimilation (DA): what is it and why is it important?

Data Assimilation (DA) is the science of optimally combining different knowledge sources that we acquire about a phenomenon modeled by various mathematical equations.These knowledge sources include:

- the mathematical model representing the physical (in the largest sense) phenomena,
- observations, also referred to as measurements or data,
- statistics on the observations and/or the model error,
- priors, probability density functions (pdf) on the modelled fields.

The goal of DA is to estimate the state of a system as it evolves in time by combining these different knowledge sources in an optimal way. In real-world problems, particularly in environmental sciences (meteorology, oceanography, hydrology, seismology, etc.), data are heterogeneous, multi-scale, and sparse both in space and time. As a consequence, they only partially represent the modeled phenomena.

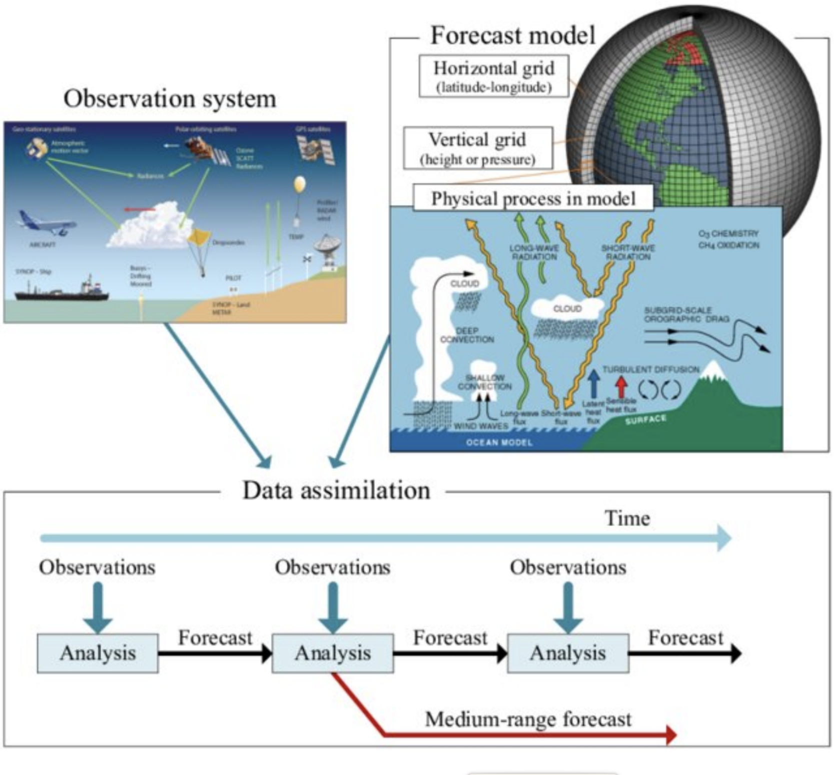


Figure 4.1: Data Assimilation basic principle for a time-dependent model (dynamic phenomena). Scheme: the sequential point of view.

The mathematical model is generally a Partial Differential Equations (PDE) system. It may also be a Stochastic Differential Equation (SDE), but SDEs are not considered in the present course. Data are generally heterogeneous (in-situ measurements, satellite images, drone videos, etc.) and of a large amount (large datasets). In this textbook, datasets are assumed to have been analyzed, pre-treated (cleaned, filtered, potentially reduced) before their assimilation into the mathematical models.

DA can be viewed as a process for solving physics-based inverse problems that incorporate various sources of uncertainty.
These uncertainties may arise from different origins, such as: - model errors (e.g., inaccuracies in the mathematical representation of the system),
- measurement errors (e.g., noise or imprecision in the observed data),
- prior information on the solution (e.g., on the initial state etc).
By integrating observational data with a mathematical model, DA seeks to provide an optimal estimate of the system's state, accounting for these diverse sources of uncertainty.

DA: what for?

Given datasets, setting up and performing a Data Assimilation (DA) process can be motivated by different goals, including:

- correcting the model output,
- calibrating the model to improve its prediction accuracy, Calibration may rely on identifying a parameter of the model, a boundary condition value (e.g., at open boundaries), or the system Initial Condition (IC) e.g., in atmospheric dynamics for weather prediction.
- identifying a physical parameter of the model. This can be for example an effective fluid viscosity, an organ conductivity in the Electrical Impedance Tomography problem mentioned in Section 1.2, or river bathymetry as in the inverse problem detailed in Section **??**.

Once calibrated, the model may be used as a physically-based interpolator between sparse data. The sparsity can be in space and/or time.

In the case of a dynamics system (a time-dependent model), once the model has been calibrated from past observations (uncertain parameters, B.C. or IC have been estimated), the model may be assumed to be more accurate for prediction, as shown in Figure 4.4.

The "traditional" DA methods mentioned above differ from purely ML approaches (e.g., Artificial Neural Networks) since they rely on a model (typically a PDE).

However, note that the Physically-Informed Neural Networks (PINNs) which are presented in a next chapter aim at combining Neural Networks (a purely Machine Learning technique) with physical models.

Distinctions and common goals between these two wide classes of methods are briefly presented in Section **??**.

4.2 The few traditional DA methods

Up to the years 2020's, DA mainly relied on two different classes of methods (so-called here traditional methods): sequential and variational ones. The choice may be driven either of the unknown parameters of the inverse problem is of large dimension or not.

4.2.1 Sequential methods (filters) estimate the state, given series of observations

The fundamental filter is the Kalman Filter (KF). The KF is optimal in the Linear Gaussian case.

In non-linear cases, one considers extensions such as: the Extended Kalma Filter (ExKF) or better, the Ensemble Kalman Filter (EnKF).

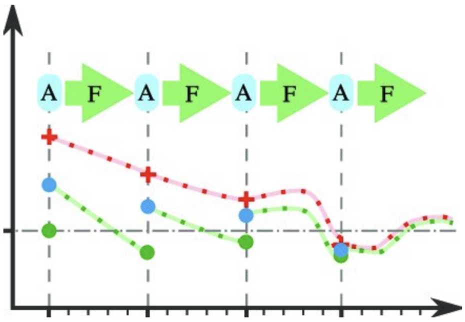


Figure 4.2: Data Assimilation method type: sequential.

4.2.2 The variational approach is based on the optimal control of the system

The Variational Data Assimilation (VDA) approach is based on the *optimal control* of the model with respect to unknown/uncertain input parameters of the model. This consists to minimize a cost function $j(u)$ measuring misfits between model outputs and measurements, while respecting the physics-based model as a constraint.

VDA enables to estimate the state of the system like filters do, but also to estimate/identify uncertain input parameters of the model.
This can be done indifferently for time-independent or time-dependent models, linear or not.

The variational approach is particularly well-suited for large dimensional non linear problems.

To model complex real-world problems, Data Assimilation (DA) must incorporate deterministic elements (based on known physics) and stochastic elements to account for uncertainties.
These uncertainties can originate from various sources: the physics model (e.g., partial differential equations) is inherently incomplete, measurements contain errors, and so on.

Another approach is based on the so-called Particle Filters (PF). This approach is not be presented in the present textbook.

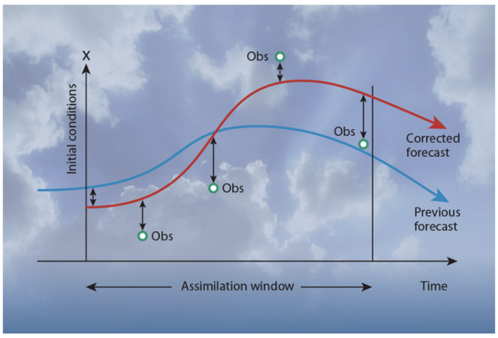


Figure 4.3: The Variational Data Assimilation (VDA) method illustrated on a 1D time-dependent model.

Each approach, sequential, variational, particle filter, possesses its own strengths, advantages, and drawbacks.

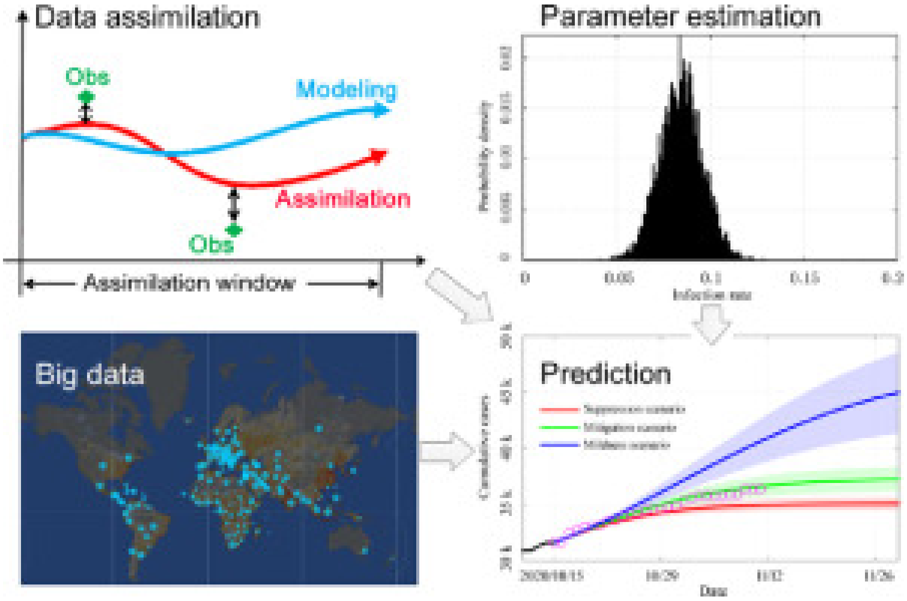
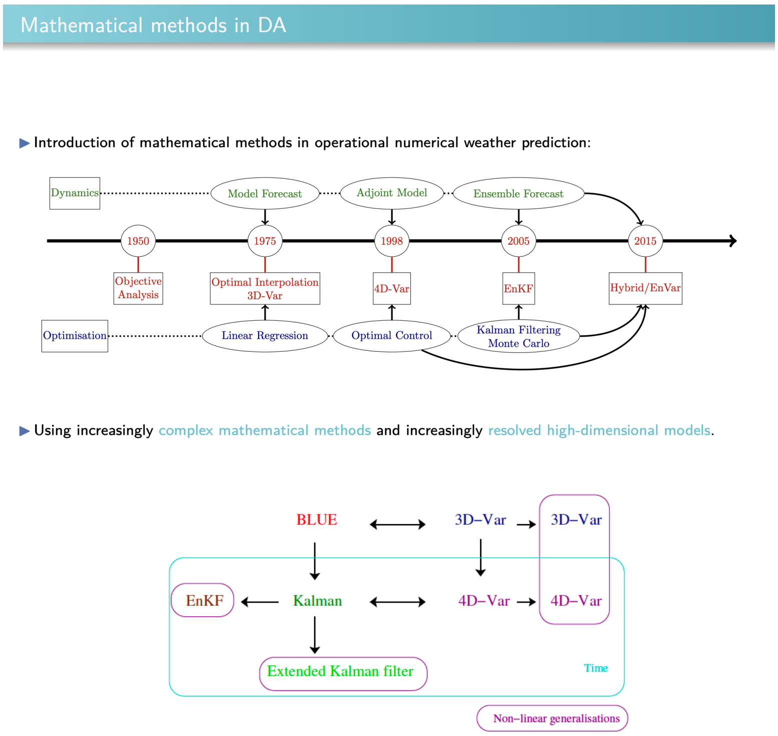


Figure 4.4: (Left) Goals of DA. 1) DA to identify an uncertain input parameter (it can be for a steady-state model). The resulting calibrated model is more accurate. It can be used as a physically-consistent interpolator between data. 2) DA to build up a predictive model (here for a time-dependent model). The model is first calibrated from past observations. Second, it is performed for prediction.

4.2.3 Time-line and summary of the traditional DA methods

The various DA methods were developed in the international weather forecast community, in institutions like the European Centre for Medium-Range Weather Forecasts (ECMWF) and the National Oceanic and Atmospheric Administration (NOAA) in the USA.



Mathematical methods in DA

▶ Introduction of mathematical methods in operational numerical weather prediction:

▶ Using increasingly complex mathematical methods and increasingly resolved high-dimensional models.

Figure 4.5: (Left) Evolution of the DA methods and their use in the weather forecast community. (Right) The different DA methods and their connections. Images source: [8]

In the 2010s, combinations of Variational Data Assimilation (VDA) and Ensemble methods have been widely adopted for large-dimensional, highly non-linear systems, such as atmospheric dynamics.

For a comprehensive overview of Data Assimilation (DA) methods, readers may refer to sources like [2] and [14].

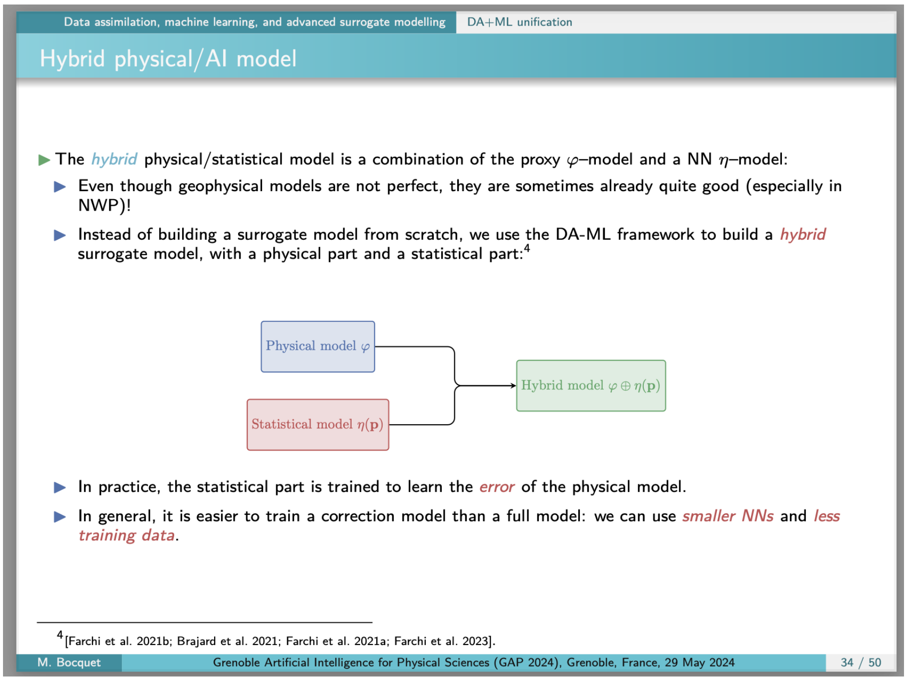
In the idealistic Linear Quadratic (LQ) cases (the model is linear and the cost function to minimize is quadratic), the VDA approach and the Kalman Filter (KF) are equivalent, assuming appropriate norms are considered.
This result will be shown later.

Furthermore, in the Linear Quadratic Gaussian (LQG) case i.e. with Gaussian fields, VDA and the KF can be elegantly interpreted within a Bayesian framework.
However, for non-linear problems (which is common in real-world scenarios), the mathematical equivalences no longer hold. Among the ”traditional” approaches, for non-linear and large-dimensional problems, the VDA formulation represents a numerically efficient (and mathematically elegant) one.

This textbook first introduces the traditional methods sequential and variational, then focuses on the VDA approach.
It also establishes a connection between DA (in particular the variational approach VDA) and the NN-based approach named Physically Informed Neural Networks (PINN).

As VDA relies on the optimal control of the physical system, this provides an opportunity to cover important aspects of optimal control (Part II of the textbook).

4.3 Today: towards hybrid DA / Deep NN methods...



Data assimilation, machine learning, and advanced surrogate modelling | DA+ML unification

Hybrid physical/AI model

- The *hybrid* physical/statistical model is a combination of the proxy φ–model and a NN η–model:
  - Even though geophysical models are not perfect, they are sometimes already quite good (especially in NWP)!
  - Instead of building a surrogate model from scratch, we use the DA-ML framework to build a *hybrid* surrogate model, with a physical part and a statistical part:[4]

Physical model φ

Statistical model $\eta(\mathbf{p})$

Hybrid model $\varphi \oplus \eta(\mathbf{p})$

  - In practice, the statistical part is trained to learn the *error* of the physical model.
  - In general, it is easier to train a correction model than a full model: we can use *smaller NNs* and *less training data*.

[4] [Farchi et al. 2021b; Brajard et al. 2021; Farchi et al. 2021a; Farchi et al. 2023].

M. Bocquet | Grenoble Artificial Intelligence for Physical Sciences (GAP 2024), Grenoble, France, 29 May 2024 | 34 / 50

Figure 4.6: Today: towards hybrid DA / ML methods... Image source: [8]

Chapter 5

DA by sequential filters

The outline of this chapter is as follows.

Contents

5.1 The Best Linear Unbiased Estimator (BLUE)

The Best Linear Unbiased Estimator (BLUE) is the simplest statistical estimator. It may be used when the underlying PDF of the measured process is unknown. It restricts the estimator to be *linear in data*. More precisely, it aims to find a linear estimator that is unbiased and has minimum variance. As a consequence, the BLUE estimator requires only the first two moments (mean and variance) of the PDF to be known.

5.1.1 A basic 1D example

Let us consider two measurements of a scalar quantity u: $z_1 = 1$ and $z_2 = 2$.
Naturally, one seeks u minimizing the following cost function:

$$j(u) = (u - z_1)^2 + (u - z_2)^2$$

The function $j(u)$ simply measures the misfit in the Euclidian norm (the 2-norm). This a standard least-square problem since u does not have to satisfy an underlying model. The solution is: $u^* = \frac{3}{2}$.

Now, let us assume that the second data represents the quantity $2u$ and not u (difference of instrument).
Then, the two measurements are: $z_1 = 1$ and $z_2 = 4$. We here seek u minimizing the cost function:

$$g(u) = (u - z_1)^2 + (2u - z_2)^2$$

In this case, the solution is $u^* = \frac{9}{5}$. This solution differs from the previous one!

This very simple example illustrates that the *the standard least-square solution depends on the norm considered in the cost function.*

It depends on the data accuracy of course too). In presence of errors, which is always the case in real-world problems, the considered norms have to take into account the measurement accuracies.

Informal definition of the BLUE

Considering the aleatory variable $\hat{u}$ defined from the data z_{obs}, the BLUE u^* is defined from the three following properties:

a) u^* is a linear function of z_{obs}.

b) u^* is unbiased (means are unchanged): $E(u^*) = u$.

c) u^* is optimal in the sense it has the smallest variance among all unbiased linear estimations.

Calculation of the BLUE for the 1D basic example

Here the observation operator is the identity, however the two measurements are assumed to contain errors. Using simple notations, we have:

$$z_i = u + \varepsilon_i, \; i = 1, 2$$

The errors of measurements ε_i are supposed to be:
- unbiased: $E(\varepsilon_i) = 0$. *(Sensors are unbiased).*

- with a known variance: $Var(\varepsilon_i) = \sigma_i,\ i = 1, 2.$ *(The sensor accuracies are known).*
- uncorrelated : $E(\varepsilon_1 \varepsilon_2) = 0.$ *(Measurements are independent hence the covariance vanishes; in addition, means vanish therefore this relation).*

Note that these assumptions are generally not satisfied in real-world problems.

By construction (Property a)), the BLUE satisfies: $u^* = a_1 z_1 + a_2 z_2$ (linear model). The coefficients a_i have to be determined. We have: $u^* = (a_1 + a_2)u + a_1\varepsilon_1 + a_2\varepsilon_2$.
By linearity of $E(\cdot)$,

$$E(u^*) = (a_1 + a_2)u + a_1 E(\varepsilon_1) + a_2 E(\varepsilon_2) = (a_1 + a_2)u$$

Property b) of the BLUE (unbiased estimator) implies that $(a_1 + a_2) = 1$ (equivalently $a_2 = (1 - a_1)$).

Recall that by definition, $Var(u^*) = E[(u^* - E(u^*))^2] = E[(u^* - u)^2]$. Therefore:

$$\begin{aligned} Var(u^*) &= E[(a_1\varepsilon_1 + a_2\varepsilon_2)^2] \\ &= a_1^2 E(\varepsilon_1^2) + 2a_1 a_2 E(\varepsilon_1\varepsilon_2) + a_2^2 E(\varepsilon_2^2) \\ &= a_1^2\sigma_1^2 + (1 - a_1)^2\sigma_2^2 \end{aligned}$$

By definition, u^* minimizes the variance (Property c)). The latter is minimal if its derivative with respect to a_1 vanishes. Therefore: $a_1 = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$.

Therefore, the BLUE reads:

$$u^* = \frac{1}{(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2})}(\frac{1}{\sigma_1^2} z_1 + \frac{1}{\sigma_2^2} z_2) \tag{5.1}$$

Note that: $Var(u^*) = \frac{\sigma_1^2\sigma_2^2}{(\sigma_1^2 + \sigma_2^2)}$. Therefore: $\frac{1}{Var(u^*)} = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right)$ and $u^* = Var(u^*)(\frac{1}{\sigma_1^2} z_1 + \frac{1}{\sigma_2^2} z_2)$.

In statistics, the inverse of a variance is called precision.

Equivalence with an optimization problem

It is easy to verify that the BLUE u^* defined by (5.1) is the unique minimum of the following quadratic cost function:

$$\boxed{j(u) = \frac{1}{2}\frac{1}{\sigma_1^2}(u - z_1)^2 + \frac{1}{2}\frac{1}{\sigma_2^2}(u - z_2)^2} \tag{5.2}$$

Indeed, we have: $j''(u) = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} = \frac{1}{Var(u^*)}$.

The Hessian of the cost function $j(u)$ (the "convexity rate" of j) equals the estimation accuracy, see Fig. 5.1.

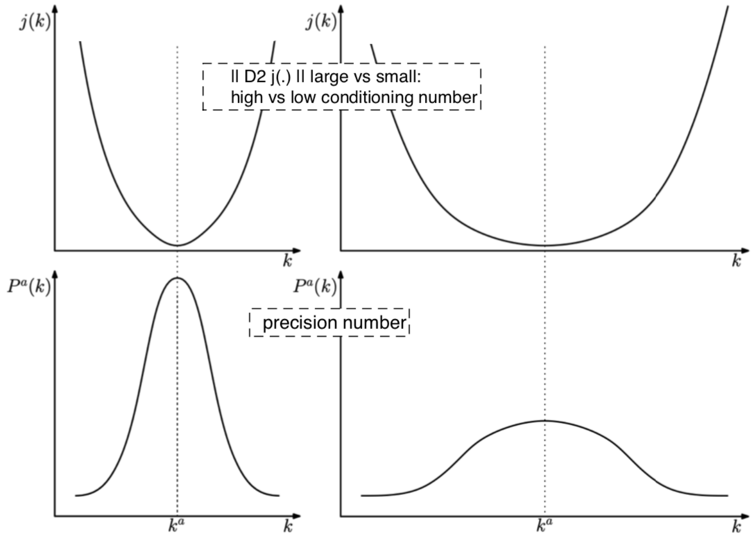


Figure 5.1: 1D case. (Up) the cost fucntiuon. (Down) The second derivative (= here, the precision).
The Hessian = the second derivative $j"(u)$ in the present 1D case.
$j"(x)$ measures the "convexity rate" of $j(u)$, therefore the estimation accuracy of the statistical estimation.

It can be quite easily shown that the BLUE calculated above (assuming unbiased measurements) minimizes the following cost function too:

$$j(u) = \frac{1}{2}(u - z_1, u - z_2) \begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}^{-1} \begin{pmatrix} u - z_1 \\ u - z_2 \end{pmatrix} = \frac{1}{2}\|u - z_{obs}\|_N^2 \tag{5.3}$$

Recall that: $\| \cdot \|_\Box^2 =< \Box \cdot, \cdot >$.

In the cost function expression above, the norm N takes into account correlations of the measurements errors: the extra diagonal terms are non vanishing.

With m observations

$$\boxed{u^* = \frac{1}{Var(u^*)}(\sum_{i=1}^{m} \frac{1}{\sigma_i^2} z_i)} \text{ with } \frac{1}{Var(u^*)} = \sum_{i=1}^{m} \frac{1}{\sigma_i^2} \tag{5.4}$$

Therefore the BLUE u^* minimizes the following cost function: $\boxed{j(u) = \frac{1}{2}\sum_{i=1}^{m}\frac{1}{\sigma_i^2}(u - z_i)^2}$.

As previously, the Hessian (here a simple scalar second derivative) defines the "convexity rate"; it measures the analysis accuracy: $j''(u) = \frac{1}{Var(u^*)}$.

With correlated errors, the extended expression $j(u) = \frac{1}{2}\|u\mathbf{1} - \mathbf{z}\|_N^2$ holds with N defined as above.

The BLUE formulated as a sequential filter in the basic 1D case

Filters are stochastic algorithms which operate recursively on *streams of (uncertain) input data* to produce a statistically optimal estimation of the underlying state.
Sequential filters are the most employed DA algorithms. The historical filter is the Kalman Filter (KF); it is presented in next section.
Let us here re-read the BLUE as a sequential filter in the basic 1d case with two observations. In this case, the BLUE expression can be re-written as:

$$u^* = \frac{\sigma_2^2 z_1 + \sigma_1^2 z_2}{\sigma_1^2 + \sigma_2^2} = z_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\right)(z_2 - z_1)$$

Let us consider that:
a) z_1 is a first estimation. It is then called the *background* or the *first guess* value.
We denote this first-guess value by u_b : $u_b \equiv z_1$.
b) z_2 is an independent observation. We denote this newly obtained observation by z : $z \equiv z_2$.

Following this point of view, the BLUE u^* reads :

$$\boxed{u^* = u_b + (\frac{\sigma_b^2}{\sigma_b^2 + \sigma_0^2})(z - u_b)} \tag{5.5}$$

The term $(z - u_b)$ is called *the innovation.* It is denoted by d, $d = (z - u_b)$.

Using the terminology of sequential data assimilation, the equation reads as follows:

$$\boxed{\text{The best estimation } u^* \text{ equals the first guess + the } \textit{gain} \text{ times the } \textit{innovation.}}$$

In the 1D case, the gain is a scalar factor, equal to $(\frac{\sigma_b^2}{\sigma_b^2+\sigma_0^2})$.

In vectorial cases, the gain is a matrix. In filter methods, defining the gain matrix is the key point. The vectorial case id developed in next section.

5.1.2 The BLUE in the general case

The linear estimation problem

All the previous calculations can be extended to the vectorial case: the system state u has n variables and one has m measurements.
Here, we seek to estimate $u = (u_1...u_n)^T \in \mathbb{R}^n$ given the measurements (observations) $z_{obs} = (z_{obs,1}, \dots, z_{obs,m})^T \in \mathbb{R}^m$. We denote by ε_{obs} the observation errors.
We have the following linear estimation problem:

$$\boxed{z_{obs} = Zu + \varepsilon_{obs} \quad u \in \mathbb{R}^n, \ z \in \mathbb{R}^m} \tag{5.6}$$

with Z a linear operator, $Z \in \mathcal{M}_{n\times m}$.

The error term ε_{obs} is assumed to be such that:
1) errors are unbiased: $E(\varepsilon_{obs}) = 0$.
2) covariances are known: $cov(\varepsilon_{obs}) = E(\varepsilon_{obs}\varepsilon_{obs}^T)$ is given.

We set:

$$R = cov(\varepsilon_{obs}) \tag{5.7}$$

The covariance matrix $cov(\varepsilon_{obs})$ is supposed to be definite therefore invertible. R^{-1} is symmetric positive definite therefore defining a norm.

In real-world problems, R is often assumed to be simply diagonal. Thus $R^{-1} = diag(\rho_{obs,1}, \dots, \rho_{obs,m})$ with $\rho_{obs,i}$ being (a-priori) precisions on the observations.

Definition

The BLUE is formally defined as follows.

Definition 5.1. *Considering the linear relation (5.6), the BLUE for u is the vector u^* such that:*

$$u^* = arg \min_{v\in\mathbb{R}^n} \left(E[(v-u)^2]\right)$$

subject to $E(v) = u$ (unbiasedness) and the constraint (5.6) of course.

Given u_b a first estimate (also called background), the error of background reads $\varepsilon_b = (u_b - u_t)$ where u_t denotes the "true" solution i.e. the exact solution satisfying $z_t = Zu_t$.
The covariance matrix $cov(\varepsilon_b)$ is assumed to be definite therefore invertible. We set:

$$B = cov(\varepsilon_b) \tag{5.8}$$

B^{-1} is symmetric positive definite therefore defining a norm.
In practice, of course that u_t and $cov(\varepsilon_b)$ are unknown. However, the following results provide useful insights.

The central result

We have:

Proposition 5.2. *Under the assumptions on the error terms ε_{obs} and ε_b above, the two statements below hold.*

1) The expression of the BLUE u^* can be explicitly derived and reads:

$$u^* = u_b + K\left(z_{obs} - Zu_b\right) \tag{5.9}$$

with u_b the first estimate (background value) and the gain matrix K defined by:

$$K = BZ^T(R + ZBZ^T)^{-1} \tag{5.10}$$

with R and B defined by (5.7) and (5.8) respectively.

2) This expression of u^* above is also the unique minimum of a quadratic cost function $j(u)$:

$$u^* = arg \min_{u \in \mathbb{R}^n} j(u) \ \ with \ j(u) = \left(\|z_{obs} - Zu\|^2_{R^{-1}} + \|u - u_b\|^2_{B^{-1}}\right) \tag{5.11}$$

The general expression (5.10) of K of course simplifies in the scalar case as in (5.5): $K = (\frac{\sigma_b^2}{\sigma_b^2+\sigma_0^2})$.

Proof. The proof of 1) can be found in the detailed online course [8], see also e.g. [20] Chapter 4.
Let us show that the first expression of K above is optimal. **ToDo: will be written ...**

The proof of 2) is much shorter. Since Z is a linear operator, the functional $j : u \in \mathbb{R}^n \mapsto j(u) \in \mathbb{R}$ is quadratic. It admits an unique minimum. This minimum is characterized by the condition: $\nabla j(u) = 0$, equivalently $j'(u) \cdot \delta u = 0$ for all $\delta u \in \mathbb{R}^n$.
We have:

$$j'(u) \cdot \delta u = 2 < R^{-1}(z_{obs} - Zu), Z\delta u > +2 < B^{-1}(u - u_b), \delta u >$$

Therefore the optimality condition $j'(u) \cdot \delta u = 0$ reads: $Z^T R^{-1}(z_{obs} - Zu) + B^{-1}(u - u_b) = 0$

$$u = u_b + BZ^T R^{-1}(Zu - z_{obs})$$

ToDo: reprendre ce calcul ...
□

Exercice 5.3. *Considering a state u of dimension 2 only, with the 2nd component measured only, retrieve the simple BLUE expression from the general expression (5.9).*

ToDo: Ecrire correction

On the norms expression

The natural norms in the present estimations problems are the ones defined from B and R (here denoted by $\Box$) as:

$$\|v - v_0\|_{\Box^{-1}} = \left(< \Box^{-1}(v - v_0), (v - v_0) >\right)^{1/2} \tag{5.12}$$

This expression of norm is called the *Mahalanobis distance*, defined as:

$$dM(v; v_0, N) = \|v - v_0\|_{N^{-1}} = \left(< N^{-1}(v - v_0), (v - v_0) >\right)^{1/2} \tag{5.13}$$

It is the natural measure in multivariate analysis. In particular, for a normal distribution $\mathcal{N}(\mu, B)$, the Gaussian PDF is determined by the Mahalanobis distance as:

$$p(u) = \frac{1}{(2\pi det(B))^{1/2}} \exp\left(-\frac{dM(u; \mu, B)}{2}\right) \tag{5.14}$$

Connection to the variational approach

The variational approach aims at making fit a model output to data by minimizing a cost function $j(u)$ with generally $j(u) = J_{obs}(u) + J_{reg}(u)$.
The expression (5.11) of the functional $j(u)$ above contain the observation misfit term $J_{obs}(u) = \|z_{obs} - Zu\|^2_{R^{-1}}$ and the misfit to background term $\|u - u_b\|^2_{B^{-1}}$ which acts as a Tikhonov's regularization term.
As a consequence, the two terms expressions $J_{obs}(u)$ and $J_{reg}(u)$ above, defined from the norms R^{-1} and B^{-1} respectively, are natural good candidates to define the cost function in the variational approach, including for non linear estimation problems.
The variational approach is studied in detail in next chapters.

5.1.3 Hessian, precision matrices

The Hessian $H_j(u)$ of the cost function $j(u)$ defined by (5.11), $H_j(u) \in \mathcal{M}_{n\times n}$, reads:

$$H_j(u) = Z^T R^{-1} Z + B^{-1} \text{ for all u } \in \mathbb{R}^n \tag{5.15}$$

Let us define the estimation error as $\varepsilon_u = (u^* - u^{true})$ and the related covariance matrix $P_u \equiv cov(\varepsilon_u) = E(\varepsilon_u \varepsilon_u^T)$.
P_u^{-1} represents the *precision* matrix.

Following the expression (5.9), we have: $\varepsilon_u = \varepsilon_b + K(\varepsilon_{obs} - Z\varepsilon_b)$.

Next, following quite long calculations, a few explicit expressions of P_u can be obtained. We refer e.g. to [8, 2]. In particular, we can show that:

$$P_u^{-1} = Z^T R^{-1} Z + B^{-1} \tag{5.16}$$

That is:
- *the precision of the estimation equals the model precision plus the background precision.*
- as already noticed in the simple scalar 1D case, *the Hessian $H_j(u)$ which measures the convexity rate of the quadratic cost function $j(u)$, also measures the estimation precision*, see Fig. 5.1.

5.1.4 Examples

The reader may consult one of the numerous well documented Python codes available on the web, e.g. the https://towardsdatascience.com webpage [1], or e.g. on https://github.com/jolange/BLUE-py.

A detailed simple example Suppose we have a data set with a parameterized PDF that depends on an unknown parameter. The BLUE estimator restricts the estimate of the parameter to be a linear combination of data samples with some weights. The goal is to find the vector that provides estimates that are unbiased and have minimum variance. The estimation problem can be solved by finding the vector that satisfies two constraints only: linearity and unbiasedness.

Below a basic Python code implementing the BLUE to estimate a parameter based on a 1D dataset.

This code first defines a (tiny) 1D dataset. It then calculates the BLUE using linear algebra. It finally plots the regression line along with the data points.

[1] https://towardsdatascience.com/linear-regression-with-ols-unbiased-consistent-blue-best-efficient-estimator-359a859f757e

```
# BLUE of a 1D dataset
import numpy as np
import matplotlib.pyplot as plt

# The 1D signal (dataset)
x = np.array([1, 2, 3, 4, 5])
y = np.array([2.1, 3.9, 6.2, 8.1, 10.1])

# Calculate the BLUE estimator (linear algebra)
X = np.vstack([np.ones(len(x)), x]).T
D = np.diag([0.1 ** 2] * len(x))
beta_hat = np.linalg.inv(X.T @ np.linalg.inv(D) @ X) @ X.T @ np.linalg.inv(D) @ y

# Calculate the precision matrix
P = np.linalg.inv(X.T @ np.linalg.inv(P) @ X)

# Plot the data and regression line
plt.scatter(x, y)
plt.plot(x, beta_hat[0] + beta_hat[1] * x)
plt.title(”BLUE Estimator”); plt.xlabel(”x”); plt.ylabel(”y”)
plt.show()
```

What happens if the model or the observation operator is non linear?

The central result previously shown holds *for a linear observation operator Z only.* In real-world problems, the estimation problem is rarely linear. Moreover $cov(\varepsilon_{obs})$ and $cov(\varepsilon_b)$ are a-priori unknown...
However for non linear estimation problems, the equivalency between the BLUE expression and the optimization problem (**??**) provide good insights to define ”optimal” R^{-1}-norm in the functional j to be minimized, see (**??**). This point is addressed later in the variational approach (VDA sections).

5.2 The Kalman Filter and extensions

Filters are stochastic algorithms which operate recursively on streams of uncertain input data to produce a statistically optimal estimation of the underlying state.
Kalman Filter (KF) has been named in honor to R.E. Kalman who has employed KF to control trajectories of the NASA Apollo program vehicles in the 60s. The KF seems to have been developed by a few different authors (Swerling (1958), Kalman (1960) and Kalman-Bucy (1961), source: Wikipedia page).

Note that *filters aim at estimating the state of the system (the model output) and not input parameters of the model* as the variational approach does (see next chapter). KF enables to improve estimation as new data is available, without recomputing from beginning. It is applied in numerous engineering domains (e.g. in the area of autonomous navigation).
KF is also called Linear Quadratic Estimator (LQE) since it optimally solves Linear Quadratic Gaussian (LQG) problems.

On sequential DA methods and KF in particular, the reader may consult e.g. the very complete book [20].

5.2.1 The linear dynamic model and observations

Filters naturally apply to *dynamical systems* since they provide a sequence of states (time series). Let us consider here a scalar linear dynamic model aiming at computing the system state u^k for all $k \geq 0$, $u^k \in \mathbb{R}^n$. The iteration k denotes here the time index. We have:

$$\boxed{u^k = Mu^{k-1} + \varepsilon_{mod}^{k-1}} \tag{5.17}$$

where M is the transition operator which is here *linear*: M is a $n \times n$ matrix.
ε_{mod} denotes the model error.
We assume that at the same time instants, observations z^k, $z^k \in \mathbb{R}^m$, are available, with:

$$\boxed{z_{obs}^k = Zu^k + \varepsilon_{obs}^k} \tag{5.18}$$

The observation operator Z is supposed to be linear too: Z denotes a $m \times n$ matrix.
Both the model errors ε_{mod} and the observation errors ε_{obs} are supposed to be Gaussian, given in $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively. They are supposed to satisfy:

$$\varepsilon_{mod}^k \sim \mathcal{N}(0, Q_k) \text{ and } \varepsilon_{obs}^k \sim \mathcal{N}(0, R_k)$$

where Q and R are the covariance matrices of the model and observation errors respectively.

5.2.2 The KF algorithm

Basic principles

At each iteration k, KF works in two steps:

`Step 1)`: the forecast (prediction) step.
A first estimation u_f^k is computed as the solution of the dynamic model (5.17).

`Step 2)`: the analysis (correction) step.
Given the newly acquired data z_{obs}^k, a corrected estimation of u^k, denoted by u_a^k, is computed.
u_f^k plays here the role of a background value.
Because of the linearity of M and Z plus the assumptions on the errors previously mentioned, *u_a^k is defined as the BLUE.*

Expressions of the KF matrices

The central KF scheme equation reads as follows: for $k \geq 1$,

$$\boxed{u_a^k = u_f^k + K^k(z_{obs}^k - Zu_f^k)} \tag{5.19}$$

with the gain matrix K^k acts as the weight of the innovation term $(z_{obs}^k - Zu_f^k)$, as in the BLUE.

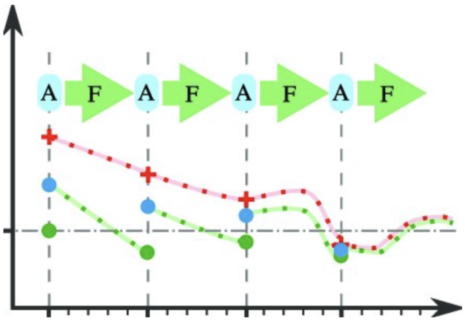


Figure 5.2: Sequential Data Assimilation method: model correction along the observation series. Cycles of Analyses (A) and Forecasts (F). (Here in a 1D case).

However, here in the expression of K^k, B is replaced (see (5.10) in the BLUE case) by the *forecast covariance errors matrix* P_f^k:

$$P_f^k = cov(\varepsilon_f^k) \text{ with } \varepsilon_f^k = (u_f^k - u_t^k)$$

Thus,

$$\boxed{K^k = P_f^k Z^T \left(R + Z P_f^k Z^T\right)^{-1}} \tag{5.20}$$

The analysis error is defined as $\varepsilon_a^k = (u_a^k - u_t^k)$. The related covariance errors matrix reads: $P_a^k = cov(\varepsilon_a^k)$. One can show that P_a^k satisfies, see e.g. [8] Chapter 2:

$$\boxed{P_a^k = (I - K^k Z) P_f^k} \tag{5.21}$$

Extreme cases: perfectly observed system / perfect model

- *Perfect model.* If the forecast errors tends to 0, that is $\|P_a^k\| \to 0$ then $K^k \to 0$ for all k.

- *Perfect data.* If the observations errors tends to 0, that is $\|R\| \to 0$ then $K^k \to Z^{-1}$ for all k.

The weighting of the innovation by the gain K may be read as follows.

- As the measurement error covariances tend to 0, the observation z^{obs} is trusted more and more while the model response u_f is trusted less and less.
- On the opposite, as the forecast error covariances tend to 0, the model response u_f is trusted more and more while the observation z^{obs} is trusted less and less.

Other simple cases In the simple case where:

- $Z = Id$,
- the covariance observation errors matrix is diagonal,
  $R \equiv \Delta_R^{obs} = diag((\sigma_1^{obs})^2, \ldots, (\sigma_m^{obs})^2)$, $(\sigma_i^{obs})^{-2}$ the precision of the i-th data,

then the gain matrix simplifies as: $K^k = P_f^k \left(\Delta_R^{obs} + P_f^k\right)^{-1}$.

Moreover, if the forecast covariance errors matrix P_f^k is diagonal and constant along the iterations,
$P_f^k = (\sigma_f)^2 Id$, $(\sigma_f)^{-2}$ the forecast precision, then the gain matrix simplifies as in the 1D simple case as:

$$K = diag\left(\frac{\sigma_f^2}{((\sigma_1^{obs})^2 + \sigma_f^2)}, \ldots, \frac{\sigma_f^2}{((\sigma_m^{obs})^2 + \sigma_f^2)}\right)$$

The KF algorithm

Initialization. The I.C. of the system state u^0 is given.
We set: $\varepsilon^0 = (u^0 - u_t)$ and $P_f^0 = cov(\varepsilon^0) = E(\varepsilon^0(\varepsilon^0)^T)$.
The error covariance matrix P_f^0 is supposed to be given too (...).
From iteration $(k-1)$ to iteration k,
1) *Analysis step.*
- Compute the *Kalman gain matrix* K^k as in (5.20).
- Deduce the analysis value u_a^k as: $\boxed{u_a^k = u_f^k + K^k(z_{obs}^k - Zu_f^k)}$.
- Compute the covariance matrix P_a^k as in (5.21).

2) *Forecast step.*
- Solve the model to obtain the forecast value u_f^{k+1}: $u_f^{k+1} = Mu_a^k$.
- Compute the covariance matrix of forecast errors P_f^{k+1} as:
$P_f^{k+1} = MP_a^k M^T + Q^{k+1}$.
(Q the covariance matrix of the model errors).

If the linear operators M and Z depend on the iteration k, the results hold and the algorithm naturally extends.

5.2.3 Examples

See one of the numerous well documented Python codes available on the web, e.g.:

- https://machinelearningspace.com/object-tracking-python/
- https://thekalmanfilter.com/kalman-filter-python-example

- https://github.com/Garima13a/Kalman-Filters

- https://arxiv.org/ftp/arxiv/papers/1204/1204.0375.pdf

A simple example

A nice simple example is proposed e.g. on towardsdatascience.com website [2]. The considered example aims at better estimating the level of a water tank given noisy sensor data.
Below an illustrative Python code based on the Pykalman library. The code first generates a time series of the water level in a reservoir (which increases linearly from 0 to 10). It then adds Gaussian noise to the water level measurements to simulate noisy measurements: this is synthetic data, randomly perturbed.
Next, one uses a basic KF to estimate the true water level from the noisy measurements.

[2] https://towardsdatascience.com/a-simple-kalman-filter-implementation-e13f75987195

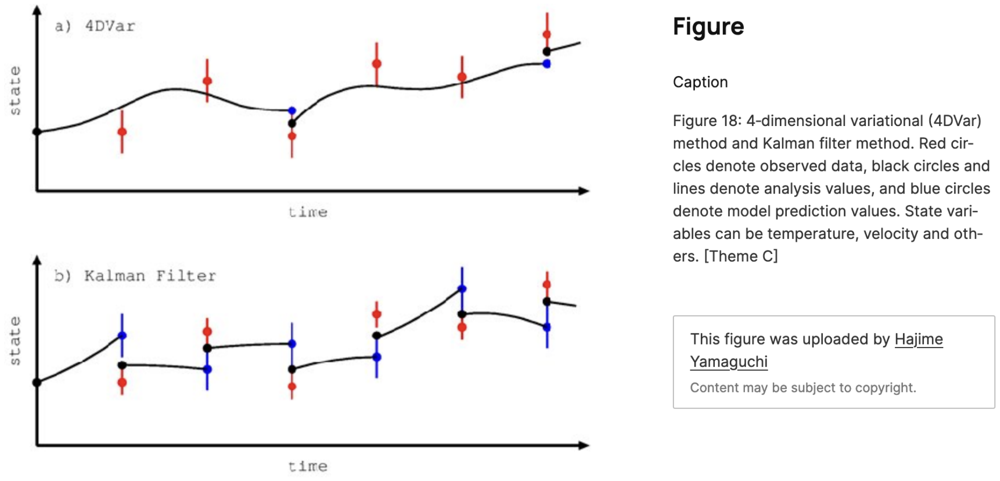
```
# Example of the use of the Kalman Filter (KF) in a very simple 1D problem
# Goal: correction of noisy measurements of a 1D signal (= e.g. water level measurements of a tank)
# Trivial linear estimation problem: z = u + epsilon_model
import numpy as np
import matplotlib.pyplot as plt

print("***************")
print("*** main.py ***")
# Generate data (synthetic data)
T =1; L = 10; npts = 100;
time = np.linspace(0, T, npts) # in IS units = seconds
data = np.linspace(0, L, npts) # + np.sin(2*np.pi/L) # the measurements (in IS units)

# observation errors: Gaussian noise
sigma_obs = 0.5
print('standard deviation of the observations errors sigma_obs = ', sigma_obs)
noise = np.random.normal(0, sigma_obs, npts) # to be tuned if necessary
noisy_data = data + noise # perturbed observations = the synthetic data

# Estimation by KF
estimated_value = np.zeros_like(data); prediction_perfect = np.zeros_like(data) # tab creation
estimated_value[0] = noisy_data[0] # 1st value of estimation = the measurement

# model error: Gaussian
sigma_f = 0.2
print('standard deviation of the model error sigma_f = ', sigma_f)

# KF gain (ref value and/or assumed to be constant)
KF_gain= sigma_f / (sigma_f + sigma_obs)
print('gain coefficient KF_gain = ',KF_gain)

# references ('"idealistic") estimations
# prediction model: z = u
perfect_values = data # perfect model values from perfect data
prediction_perfect_model = noisy_data # perfect model values from noisy data

# estimation by KF
for i in range(1, len(time)):
    # prediction model: z = (Identity + eps_model)(u)
    predicted_value = estimated_value[i - 1] + np.random.normal(0, sigma_f)

    # analysis step
    gain = KF_gain # constant gain (here a scalar value)
    innovation = noisy_data[i] - predicted_value
    estimated_value[i] = predicted_value + gain * innovation # the analysis value

# Plots
plt.plot(time, noisy_data, 'o', label='measurements (= the data)')
plt.plot(time, estimated_value, 'k', label='estimation by KF')
plt.plot(time, prediction_perfect_model, 'c', label='prediction based on the perfect model from noisy data
plt.plot(time, perfect_values, '--r', label='perfect values')
plt.xlabel('time'); plt.ylabel('values')
plt.legend(); plt.show()
```

5.2.4 Pros and cons of KF

⊕ Considering a linear model M and a linear observations operator Z, assuming that the covariances $cov(\varepsilon_{mod})$ and $cov(\varepsilon_{obs})$ are Gaussian, KF is the optimal sequential method, see e.g. [14].

⊕ At each iteration k, the new estimation u_a^k depends on the previous time step state u^{k-1} and the current measurement z^k only: no additional past information is required, see Fig.

5.3.



Figure

Caption

Figure 18: 4-dimensional variational (4DVar) method and Kalman filter method. Red circles denote observed data, black circles and lines denote analysis values, and blue circles denote model prediction values. State variables can be temperature, velocity and others. [Theme C]

This figure was uploaded by Hajime Yamaguchi

Content may be subject to copyright.

Figure 5.3: (Up) VDA algorithm (Down) KF algorithm. Figure extracted from [?].

⊖ The KF scheme (5.19) is bearly affordable for large dimension general problems (dim(u) >>). Indeed, to compute the gain matrix K, see (5.10), one needs to invert the matrix $(R + ZBZ^T)$.
Without specific properties of the involved matrices, in the case of n and/or m large, this requires high computational ressources.

⊖ The basic version of the KF does not apply to non-linear models.
In non-linear cases, KF can be extended by linearizing the ”transition” operator M at each time step: this is the idea of the Extended Kalman Filter (ExKF).
However, the ExKF is not optimal anymore. Moreover, its use is limited too by its high computational requirements as the KF.

5.2.5 Extension to non-linear models and/or large dimensional problems: Ensemble KF (EnKF) and hybrid approaches

For non-linear large dimensional cases, one uses variants of KF such as for example the SEEK filter, [?].
Also, one classically uses the Ensemble Kalman Filter (EnKF). In a nutshell, EnKF consists to perform a Monte-Carlo algorithm to estimate the covariance errors matrices. EnKF have shown excellent results even for high-dimensional problems ($O(10^8)$ and more), see e.g. [20, 13].
In short, the Ensemble Kalman Filter (EnKF) is a recursive filter that is used to estimate the state of a system as the basic KF. However it is suitable for large dimensional problems. The EnKF is a Monte Carlo implementation of the Bayesian update problem, which involves updating the probability density function (PDF) of the state of the modeled system after taking into account the data likelihood.

One advantage of EnKF is that advancing the PDF in time is achieved by simply advancing each member of the ensemble.

It is worth noticing that the EnKF makes the assumption that all PDF involved are Gaussian. As already mentioned, this assumption is a-priori not true for non-linear problems...

In summary,

KF The KF is an algorithm that estimates the state of a linear dynamic system based on noisy measurements. It uses a recursive process to update the state estimate as new measurements become available. At each iterate, the estimator relies on the BLUE. The KF is widely used in various fields, including engineering, economics, and computer science. However, the KF is restricted to linear estimation problems of relatively small dimensions or particular large dimension problems.

EnKF The EnKF is an extension of the KF that addresses the limitations of the KF for large dimensional and non linear systems. It uses an ensemble of state vectors to represent the probability distribution of the system state. The EnKF updates the ensemble members based on observations and propagates them forward in time using a numerical model. It is widely employed in geosciences but not only.

VDA As it will studied now, the VDA method is based on another approach: it aims to find the optimal estimate of the system state by minimizing the cost function $j(u)$.
It then relies on an optimization algorithm which iteratively adjusts the state (model prediction) $y(u)$ based on the observations z_{obs}.
The variational approach is a good option to address large dimensional problems (m=dim(u) >>) and non linear problems (M and/or Z non linear). It is widely employed in geosciences but not only.

Note that the current state-the-art for operational large dimensional and complex multi-physics non-linear models consists to combine VDA with EnKF.
Such hybrid approaches have been developed in particular in operational weather forecast centers like ECMWF or NOAA for example.

In the linear Gaussian case (this has to be clarified), each of these methods can be nicely related to the estimations obtained using a Bayesian analysis.

To study the non-VDA DA methods in details, the reader may consult e.g. [20, 2, 14] and one of the excellent material available online e.g. http://www.cs.unc.edu/ welch/kalman, [9] etc.

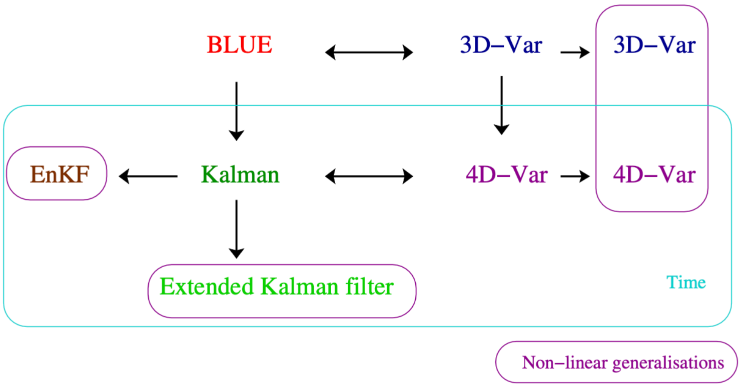


Figure 5.4: The classical DA methods. Image extracted from [8]

Chapter 6

Bayesian inferences

The outline of this chapter is as follows.

Contents

6.1 Bayesian analysis

Rev. Thomas Bayes (1702-1761), English statistician and philosopher. The Bayes's law, also called Bayes's theorem, and the Bayesian interpretation of probability was independently developed by P.-S. Laplace (1749-1827).

Bayesian analysis is a method of statistical inference that allows one to combine prior information about a parameter with evidence[1] from information contained in a dataset (a sample). The method involves specifying a *prior* probability distribution for the parameter of interest,

[1] evidence: something that increases the probability of a supported hypothesis

obtaining evidence. By combining the prior distribution with the evidence and using the Bayes theorem, a *posterior* probability distribution for the parameter is obtained.
Then, Bayesian inference provides an approach to tackle inverse problems into a probabilistic framework. The resulting posterior provides rich information on the sought parameter(s), that is the solution of the inverse problem. However, Bayesian inference is not suitable to large dimensional problems due to the so-called "curse of dimensionality".
For a complete study on Bayesian analysis, the reader may consult the phenomenal book [21, 25].

6.1.1 Founding calculations

Problem statement

Let u be the parameter to be estimated given M observations $z_{obs} = (z_{obs,1}, \dots, z_{obs,M})$, with:

$$\boxed{z_{obs} = \mathcal{M}(u) + \varepsilon} \tag{6.1}$$

with $\mathcal{M}$ denoting a non linear operator and ε the error term, $\varepsilon = (\varepsilon_{obs} + \varepsilon_{mod})$.
ε_{obs} represents the observation error and ε_{mod} represents structural model errors. On contrary to the observation error, the model error has to be inferred during the model estimation, together with the model parameter u.
For a sake of simplicity, it is assumed here that: $\varepsilon_{mod} = 0$.

The Bayes law

Let $p(u)$ be the prior distribution of u. Let $p(z_{obs}|u)$ be the probability of z_{obs} given u: it is the *likelihood* resulting from the (direct) model $\mathcal{M}$.
The joint probability density of u and z_{obs} reads in terms of the conditional densities as:

$$p(u, z_{obs}) = p(u|z_{obs})p(z_{obs}) = p(z_{obs}|u)p(u)$$

This provides the Bayes's law:

$$\boxed{p(u|z_{obs}) = \frac{p(z_{obs}|u)}{p(z_{obs})}p(u)} \tag{6.2}$$

This relation can read as:

$$\text{Posterior} \propto (\text{ Likelihood} \times \text{Prior })$$

The denominator $p(z_{obs})$ is simply a normalizing constant. (It is sometimes called the evidence). Its value may be obtained by integrating over all u: $p(z_{obs}) = \int p(y|u)\ p(u)du$. This value may be chosen such that the total mass of the posterior distribution $p(u|z_{obs})$ equals 1 too.

$p(u|z_{obs})$ is the *posterior* distribution (a-posteriori density).

In the present inverse problem context, the posterior $p(u|z_{obs})$ contain all information on the sough quantity u (given z_{obs}).

In practice, values of interest are generaly the most probable value $u^* = arg \max_u p(u|z_{obs})$ called the Maximum A-Posteriori (MAP), or the posterior mean $\bar{u} = \text{mean}(p(u|z_{obs}))$, or quantiles of $p(u|z_{obs})$.

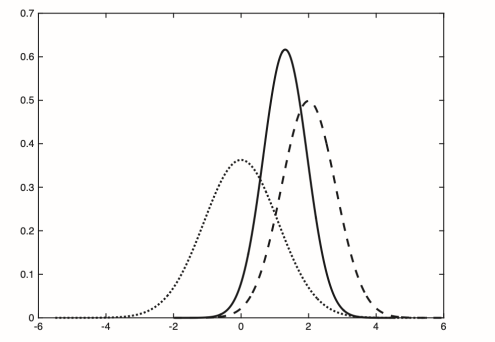


Figure 6.1: The distributions in a trivial scalar/univariate Gaussian case: prior (dotted), likelihood (dashed) and the resulting posterior (solid) Gaussian distributions. Bayes' law: Posterior $\propto$ Prior $\times$ Likelihood. Figure extracted from [2].

Remark 6.1.

- *The likelihood $p(z_{obs}|u)$ can be numerically approximated by a Monte-Carlo method or a MCMC method (use of Markov Chains to perform Monte Carlo estimations) by evaluating the likelihood, therefore solving the direct model $\mathcal{M}$, a very large number of times (say $O(10^4)$ for a one-dimensional parameter u). This implies to tackle an extremely low CPU-time consuming model $\mathcal{M}(u)$.*
- *If a lot of data are available, the choice of the prior doesn't matter so much.*
  On the contrary, if not so much data are available, the choice of a relevant prior (the background information) becomes crucial since the result (the posterior) highly depends on the prior...
- *For large scale real-world problems, the priors may be quite subjective, as a consequence the resulting posteriors become subjective too...*
- *Since the posterior $p(u|z_{obs})$ results from the data z_{obs}, it is expected that it is less variable than the prior $p(u)$....*

For an arbitrary prior distribution, generally no analytical solution of the posterior distribution is known, only approximations may be obtained.

On the contrary, in the *Linear Gaussian case* i.e. the prior $p(u)$ is Gaussian and the model operator $\mathcal{M}$ is linear, the likelihood $p(z_{obs}|u)$ is Gaussian and the posterior distribution $p(u|z_{obs})$ is Gaussian too.
In this case, the expression of $p(u|z_{obs})$ can be calculated. This expression is presented in a next paragraph.

6.1.2 The most probable parameter u

As estimator let us consider *the most probable parameter value u.*
Moreover, we introduce here a background value u_b, which is an additional prior of the estimation problem.
Given the observations z_{obs}, given the background value u_b, the most probable parameter satisfies:

$$\boxed{u^* = arg \max_u \left(p(u|z_{obs} \text{ and } u_b)\right)} \tag{6.3}$$

Since the function $(-\log)(\cdot)$ is monotonic decreasing therefore convex, the optimization problem above can be equivalently written by minimizing the following functional:

$$\boxed{j(u) = -\log(p(u|z_{obs} \text{ and } u_b)) + c} \tag{6.4}$$

with c denoting any constant value. Then the most probable parameter u satisfies:

$$\boxed{u^* = arg \min_u j(u)}$$

Assuming that the observation errors and the background errors are uncorrelated (this assumption is a-priori satisfied), we get:

$$p(u|z_{obs} \text{ and } u_b) = p(u|z_{obs})\ p(u|u_b)$$

Using the Bayes law (6.2), it follows:

$$j(u) = -\log p(u|z_{obs}) - \log p(u|u_b) + \text{ c} \tag{6.5}$$

for any constant c.

The calculations above are valid for any distributions $p(u|z_{obs})$ and $p(u|u_b)$.
Next, if considering Gaussian distributions (Probability Density Functions -PDFs-), the choice of the log function in the definition of $j(u)$ turns out to be judicious...

6.2 Assuming Gaussian PDFs

6.2.1 Scalar / univariate case

For sake of simplicity, we here consider the *scalar / univariate case*: the parameter u is a scalar function.

The prior distribution and likelihood

On the prior distribution $p(u|u_b)$.
Let us assume $p(u|u_b)$ is Gaussian: $p(u|u_b) \sim \mathcal{N}(\overline{u_b}, \sigma_u^2)$, that is

$$p(u|u_b) = \frac{1}{(2\pi)^{1/2}\sigma_u} \exp\left(-\frac{1}{2\sigma_u^2}(u - \overline{u_b})^2\right) \tag{6.6}$$

On the likelihood $p(z_{obs}|u)$.
Recall that given a Gaussian prior $p(u)$, *if* the model operator $\mathcal{M}$ is *linear* then the likelihood $p(z_{obs}|u)$ is Gaussian. On the contrary if $\mathcal{M}(\sqcap)$ is non linear then the likelihood $p(z_{obs}|u)$ is non Gaussian.

We get here M observations : $z_{obs} = (z_{obs,1}, \ldots, z_{obs,M})$, with $\varepsilon_{obs} \sim \mathcal{N}(0, \sigma_{obs})$.
Let us assume that $p(z_{obs}|u)$ is Gaussian (whatever if $\mathcal{M}$ is linear or not): $p(z_{obs}|u) \sim \mathcal{N}(\overline{z_{obs}(\mathcal{M};u)}, \sigma_{obs}^2)$.
Moreover we assume that the M observations are all independent. Then, we get:

$$p(z_{obs}|u) = \Pi_{m=1}^{M} \frac{1}{(2\pi)^{1/2}\sigma_{obs}} \exp\left(-\frac{1}{2\sigma_{obs}^2}(z_{obs,m} - \overline{z_{obs}(\mathcal{M};u)})^2\right)$$

that is:

$$p(z_{obs}|u) \propto \exp\left(-\frac{1}{2\sigma_{obs}^2}\sum_{m=1}^{M}(z_{obs,m} - \overline{z_{obs}(\mathcal{M};u)})^2\right) \tag{6.7}$$

Resulting posterior distribution

If both $p(u|u_b)$ and $p(z_{obs}|u)$ are Gaussian then the posterior $p(u|z_{obs})$ is Gaussian too, as the product of two Gaussians.
Indeed, by applying the Bayes law (6.2), the posterior reads as:

$$p(u|z_{obs}) \propto \exp\left(-\frac{1}{2\sigma_u^2}(u - \overline{u_b})^2 - \frac{1}{2\sigma_{obs}^2}\sum_{m=1}^{M}(z_{obs,m} - \overline{z_{obs}(\mathcal{M};u)})^2\right) \tag{6.8}$$

which is equivalent too (see e.g. [3] Chap. 3 for detailed calculations):

$$p(u|z_{obs}) \propto \exp\left(-\frac{1}{2\sigma_p^2}(u - \overline{u_p})^2\right) \tag{6.9}$$

$$\text{with } \overline{u_p} = \sigma_p^2\left(\frac{u_b}{\sigma_u^2} + \sum_{m=1}^{M}\frac{z_{obs,m}}{\sigma_{obs}^2}\right) \text{ and } \sigma_p^2 = \left(\frac{1}{\sigma_u^2} + \frac{M}{\sigma_{obs}^2}\right)^{-1} \tag{6.10}$$

6.2.2 Vectorial / multivariate case*

This is a "to go further" section.

In the vectorial / multivariate case, u is a vectorial function. A Gaussian distribution is described by a mean which is vectorial and a covariance matrix. The calculation principles remain the same but a bit more complex.
The Gaussian prior $p(u)$ reads as $p(u) \sim \mathcal{N}(\overline{u_b}, B^{-1})$ with B a given invertible covariance matrix[2], thus:

$$p(u) = \frac{1}{(2\pi)^{n/2}\sigma_u} \exp\left(-\frac{1}{2} \|u - \overline{u_b}\|^2_{B^{-1}} \right) \tag{6.11}$$

with $\sigma_u^2 = det(B)$.

The likelihood is supposed to be Gaussian: $p(z_{obs}|u) \sim \mathcal{N}(\overline{z_{obs}}, R^{-1})$, with R an invertible covariance matrix. The M observations are supposed to all independent, therefore:

$$p(z_{obs}|u) \propto \exp\left(\sum_{m=1}^{M} \|z_{obs,m} - \overline{z_{obs}}\|^2_{R^{-1}} \right) \tag{6.12}$$

Like in the scalar case, by applying the Bayes law (6.2) the posterior distribution $p(u|z_{obs})$ is Gaussian as the product of two Gaussians whose expression satisfies (see e.g. [3] for more details on the calculations):

$$p(u|z_{obs}) \propto \exp\left(-\frac{1}{2} \|u - \overline{u_p}\|^2_{P^{-1}} \right) \tag{6.13}$$

with the (vectorial) mean value $\overline{u_p}$ which satisfies:

$$\overline{u_p} = P^{-1} \left(B^{-1}\overline{u_b} + Z^T R^{-1} z_{obs} \right) \tag{6.14}$$

The posterior covariance matrix P satisfies, see e.g. [3]:

$$P^{-1} = (B^{-1} + Z^T R^{-1} Z) \tag{6.15}$$

One can show that $P^{-1} = (I - KZ)B$ with K the gain matrix which can be calculated, see e.g. [3].

6.3 The Maximum A-Posteriori (MAP) in the Gaussian case

Recall (6.3)-(6.5): the most probable estimation (=the MAP) satisfies:

$$u^* = arg \max_u p(u|z_{obs}) = arg \min_u j(u) \tag{6.16}$$

by setting adequate constant values.

[2] The inverse of a covariance matrix (if existing) is a so-called precision matrix.

By using the Gaussian forms distributions, see (6.6)(6.7) in the scalar case, it follows that:

$$\boxed{u^* = arg \min_u j(u) \text{ with } j(u) = \frac{1}{2}\|Mu - \overline{z_{obs}(\mathcal{M};u)}\|^2_{\sigma_{obs}^{-1}} + \frac{1}{2}\|u - \overline{u_b}\|^2_{\sigma_u^{-1}}} \quad (6.17)$$

In summary, computing the most probable posterior value (= the MAP) involves maximizing a product. Next, by considering the log-likelihood function and under Gaussian assumptions leads to the minimization of the quadratic cost function $j(u)$ defined in (6.17).

Remark 6.2. *The term $\|u-\overline{u_b}\|^2_\square$ in the definition of $j(u)$, see (6.17) can be read as a Tikhonov regularization term, see Section 2.2.2. In the present probabilistic point of view, this "regularization" term corresponds to the prior $p(u|u_b)$ whose is assumed to be Gaussian.*

6.4 Numerical computations

6.4.1 Algorithm

In the computational point of view, given a Gaussian prior $p(u)$ and assuming a Gaussian likelihood $p(z_{obs}|u)$, the posterior $p(u|z_{obs})$ is numerically approximated as follows.

- Define a sufficiently fine grid (sampling) of the parameter space $\mathcal{U}$, $\mathcal{U} \subset \mathbb{R}^n$.
- Compute an approximation of the prior $p(u)$ by evaluating N_u values $p(u_n)$, $n = 1, \dots, N_u$.
  We have: $N_u = O(n_0^n)$ with n_0 large enough, that is $n_0 = O(10)$ at least ?...
- Compute an approximation of the likelihood distribution $p(z_{obs}|u)$ as follows.
  - Perform the N_u model outputs $\mathcal{M}(u_n)$, $n = 1, \dots, N_u$,
  - Given a fixed Gaussian likelihood form as $\mathcal{N}(\overline{z_{obs}(\mathcal{M};u)}, \sigma_{obs})$, evaluate the N_u probabilities $p(z_{obs}|u_n)$ as:

$$p(z_{obs}|u_n) = \frac{1}{(2\pi)^{1/2}\sigma_{obs}} \exp\left(-\frac{1}{2\sigma_{obs}^2}\left(\mathcal{M}(u_n) - \overline{z_{obs}(\mathcal{M};u)}\right)^2\right) \text{ for } n = 1, \dots, N_u,$$

  - Evaluate the approximation of the likelihood $p(z_{obs}|u)$ as:

$$p(z_{obs}|u) \approx \Pi_{n=1}^{N_u} p(z_{obs}|u_n)$$

- Deduce the approximation of the posterior distribution $p(u|z_{obs})$ as:

$$p(u|z_{obs}) \propto \text{ Prior } p(u) \times \text{ Likelihood } p(z_{obs}|u)$$

An alternative to explore the parameter space $\mathcal{U}$ is to employ a MCMC method like the Metropolis-Hastings algorithm. However, MCMC algorithms still require huge number of model output evaluations (let us say $O(10^4)$ and more).

6.4.2 Pros and cons

$\oplus$ The Bayesian analysis provides the richest information one can expect on the tackled inverse problem: the complete posterior distribution $p(u|z_{obs})$ from which one can next deduce the most probable value (= the MAP), the posterior mean and quantiles.

$\oplus$ The algorithm can be applied by performing the model $\mathcal{M}(\cdot)$ as a black box only. In this sense, it is a non-intrusive method.

$\ominus$ The approach is not tractable neither for CPU-time consuming models $\mathcal{M}(\cdot)$ nor for non tiny parameter dimension n.

$\ominus$ The posterior directly results from both the prior and the likelihood. The latter can be unknown...
Moreover, even if the prior distribution of u is supposed to be Gaussian (which may be a reasonable assumption), the resulting likelihood $p(z_{obs}|u)$ is not Gaussian in the case of a non linear operator $\mathcal{M}$.

6.5 Examples

The reader can find numerous well documented examples with corresponding Python codes available on the web, e.g. on the https://towardsdatascience.com web site [3], [4].

A detailed simple example Below is presented a Python code computing a simple Bayesian analysis to estimate a scalar parameter u.
First, synthetic data are generated with a given value of u (0.6 in this case).
The prior distribution for u is defined: a Beta distribution with $\alpha = \beta = 1$.
The likelihood of the observed data z_{obs} given u is specified: it is supposed to be a Bernoulli distribution.
Finally, a Markov chain Monte Carlo (MCMC) sampling is employed to obtain samples from the posterior distribution of u.
The prior, likelihood and the resulting estimated posterior are plotted, as well as the trace of the MCMC algorithm.

[3] https://towardsdatascience.com/how-to-use-bayesian-inference-for-predictions-in-python-4de5d0bc84f3

[4] https://towardsdatascience.com/estimating-probabilities-with-bayesian-modeling-in-python-7144be007815

```
# Example of a 1D (scalar function) Bayesian analysis
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns # plots
import pymc3 as pm # Bayesian modeling and Probabilistic ML relying on MCMC

# Generate synthetic data: binomial
np.random.seed(0)
data = np.random.binomial(n=1, p=0.6, size=100)

# Define the probabilistic "model"
with pm.Model() as first_model:
    u = pm.Beta('u', alpha=1., beta=1.) # the prior = beta distribution
    z = pm.Bernoulli('z', p=u, observed=data) # the likelihood = Bernoulli distribution

# to draw samples of the posterior (trace of the MCMC algorithm)
    trace = pm.sample(1000, random_seed=123)
    #before drawing these 'real' samples, PyMC3 lets the chain converge to the distribution of your model
    #once this phase is complete, it starts drawing from the distribution. The number of tuning iterations

# Plots
# posterior distribution
pm.plot_posterior(trace, var_names=['u'], credible_interval=0.95, label='posterior')
# prior distribution
sns.distplot(np.random.beta(1, 1, 1000), hist=False, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4}, label='prior')
# likelihood distribution
sns.distplot(np.random.binomial(n=1, p=np.mean(data), size=1000), hist=False, kde=True,
             bins=int(180/5), color = 'darkgreen',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4}, label='likelihood')

plt.xlabel('u'); plt.ylabel('Density'); plt.title('Distributions')
plt.show()

# Plot of the MCMC sampling process
pm.traceplot(trace)
plt.show()

# Summary of the MCMC output information
%pm.summary(trace).round(2)
```

Chapter 7

Equivalences & completeness between methods

7.1 Equivalence between the traditional DA methods in the Linear Quadratic Gaussian (LQG) case

Let us recall the linear estimation problem considered in the previous Bayesian analysis: estimating u which satisfies $z_{obs} = (\mathcal{M}(u) + \varepsilon)$, with $\mathcal{M}$ denoting a non linear operator, ε the errors term $\varepsilon = (\varepsilon_{obs} + \varepsilon_{mod})$.
For a sake of simplicity, it is assumed here that $\varepsilon_{mod} = 0$.
Moreover, we assume that:

- the error is Gaussian, $\varepsilon_{obs} \sim \mathcal{N}(0, R^{-1})$,
- the model $\mathcal{M}$ is linear: $Z \equiv M$, $Z \in \mathcal{M}_{n\times m}$.

Under the assumptions above, we solve the estimation problem: find $u \in \mathbb{R}^n$ satisfying

$$z_{obs} = Zu + \varepsilon_{obs} \tag{7.1}$$

7.1.1 The equivalence

It has been previously demonstrated that:

- the Best Linear Unbiased Estimator (BLUE) (the simplest statistic estimator) is equivalent to minimize the following quadratic functional (see Prop. 5.2):

$$j(u) = \left(\|z_{obs} - Zu\|^2_{R^{-1}} + \|u - u_b\|^2_{B^{-1}}\right) \tag{7.2}$$

  where the norms R^{-1} and B^{-1} are defined as $R = (cov(\varepsilon_{obs}))$ and $B = (cov(\varepsilon_b))$ respectively.

- if the prior distribution $p(u)$ is Gaussian, then the most probable solution u^* (= the MAP) coincides with the BLUE, provided we use the same B^{-1} and R^{-1} norms as above.
  The MAP is nothing else than the (unique) optimum of the Quadratic function $j(u)$ defined by (7.2), considering the appropriate covariances matrices R and B[1].

Moreover, we recall that the *VDA method* aims at minimizing the functional $j(u)$ defined by (7.2).

In conclusion, in the LQG case, the three approaches, namely the deterministic VDA, the statistic BLUE / sequential KF and the MAP, mathematically yield the same estimation u^*!

In other words, the VDA solution and the sequential KF estimation are fully interpretable through Bayesian analysis.

However, each approach lead to different algorithms, thus coming with distinct advantages and drawbacks.

It is important to note that this mathematical equivalence holds true only in the LQG case, which is a-priori not the case in real-world problems. Furthermore in practice, the error covariances matrices or the likelihood are often unknown.
This result in the LQG case, however, provides a strong guideline to interpretate the results in the LQG cases but also to address non linear problems.

Note that the KF can be derived from the Bayesian analysis too, see e.g. [2] Chapter 3.

7.1.2 Case of large dimensional problems

$\ominus$ In the case of large dimensional problems, i.e. with m =dim(u) >>, the Bayesian analyses and the KF-based filters are barely tractable since too highly CPU-time and memory consuming.

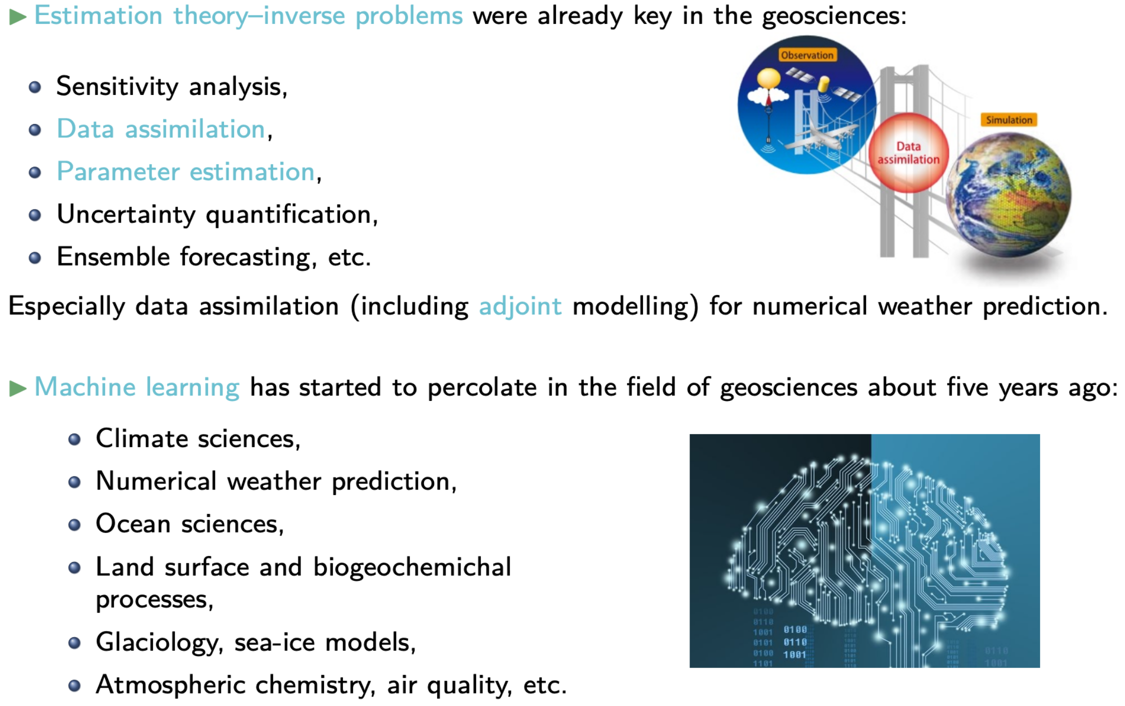
$\oplus$ On the contrary, the VDA method "naturally" extends to large dimensional cases and to non linear models.

Indeed, the corresponding algorithms (3D-Var, 4D-Var and so on, see subsequent chapters) are well-suited for large dimensional and non linear problems since based on local gradient-based optimisation algorithms.

[1] In the LQ case, the cost function $j(u)$ is strictly convex therefore admitting a unique minimum. This point is mathematically shown in a subsequent chapter.

⊖ However, uncertainties are not provided by the VDA method, on contrary to the Bayesian approach.

7.2 Completeness between the traditional DA methods and Machine Learning



▶ Estimation theory–inverse problems were already key in the geosciences:

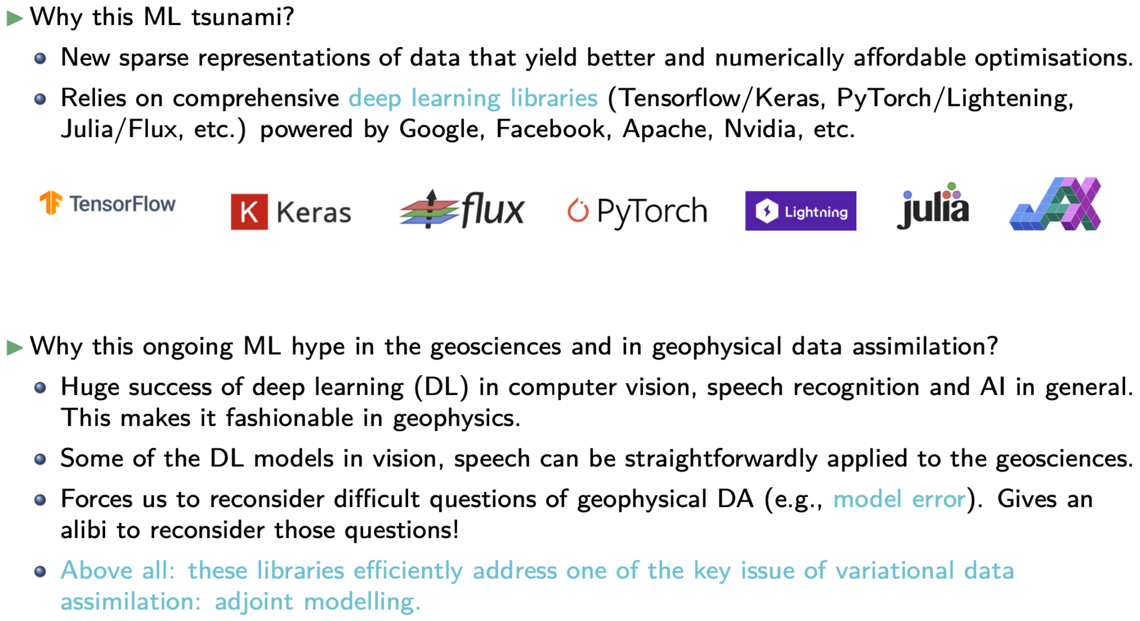
- Sensitivity analysis,
- Data assimilation,
- Parameter estimation,
- Uncertainty quantification,
- Ensemble forecasting, etc.

Especially data assimilation (including adjoint modelling) for numerical weather prediction.

▶ Machine learning has started to percolate in the field of geosciences about five years ago:

- Climate sciences,
- Numerical weather prediction,
- Ocean sciences,
- Land surface and biogeochemichal processes,
- Glaciology, sea-ice models,
- Atmospheric chemistry, air quality, etc.

Figure 7.1: Towards hybrid DA / ML methods... Image source: [8]



▶ Why this ML tsunami?

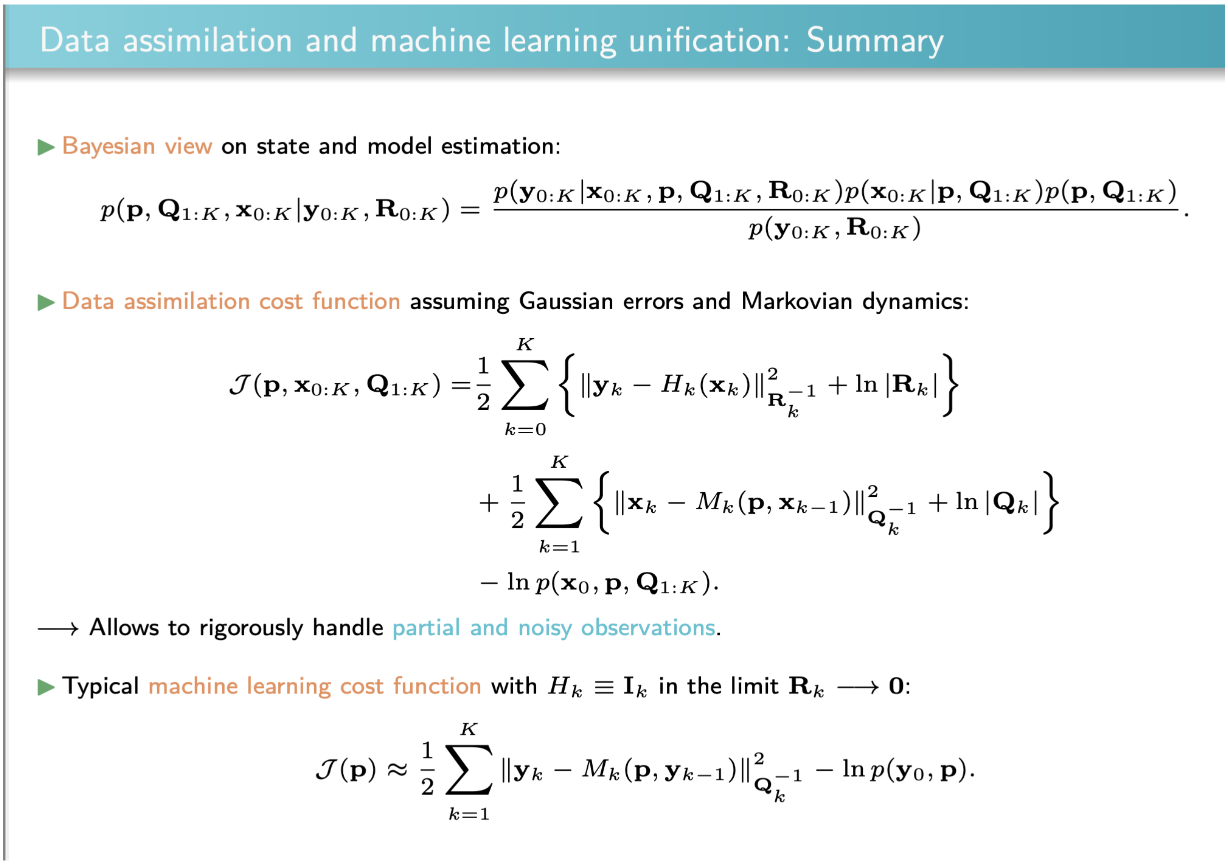
- New sparse representations of data that yield better and numerically affordable optimisations.
- Relies on comprehensive deep learning libraries (Tensorflow/Keras, PyTorch/Lightening, Julia/Flux, etc.) powered by Google, Facebook, Apache, Nvidia, etc.

TensorFlow
Keras
flux
PyTorch
Lightning
julia

▶ Why this ongoing ML hype in the geosciences and in geophysical data assimilation?

- Huge success of deep learning (DL) in computer vision, speech recognition and AI in general. This makes it fashionable in geophysics.
- Some of the DL models in vision, speech can be straightforwardly applied to the geosciences.
- Forces us to reconsider difficult questions of geophysical DA (e.g., model error). Gives an alibi to reconsider those questions!
- Above all: these libraries efficiently address one of the key issue of variational data assimilation: adjoint modelling.

Figure 7.2: Towards hybrid DA / ML methods... Image source: [8]



Data assimilation and machine learning unification: Summary

▶ Bayesian view on state and model estimation:

$$p(\mathbf{p}, \mathbf{Q}_{1:K}, \mathbf{x}_{0:K} | \mathbf{y}_{0:K}, \mathbf{R}_{0:K}) = \frac{p(\mathbf{y}_{0:K} | \mathbf{x}_{0:K}, \mathbf{p}, \mathbf{Q}_{1:K}, \mathbf{R}_{0:K}) p(\mathbf{x}_{0:K} | \mathbf{p}, \mathbf{Q}_{1:K}) p(\mathbf{p}, \mathbf{Q}_{1:K})}{p(\mathbf{y}_{0:K}, \mathbf{R}_{0:K})}.$$

▶ Data assimilation cost function assuming Gaussian errors and Markovian dynamics:

$$\begin{aligned} \mathcal{J}(\mathbf{p}, \mathbf{x}_{0:K}, \mathbf{Q}_{1:K}) =& \frac{1}{2} \sum_{k=0}^{K} \left\{ \|\mathbf{y}_k - H_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \ln |\mathbf{R}_k| \right\} \\ &+ \frac{1}{2} \sum_{k=1}^{K} \left\{ \|\mathbf{x}_k - M_k(\mathbf{p}, \mathbf{x}_{k-1})\|^2_{\mathbf{Q}_k^{-1}} + \ln |\mathbf{Q}_k| \right\} \\ &- \ln p(\mathbf{x}_0, \mathbf{p}, \mathbf{Q}_{1:K}). \end{aligned}$$

⟶ Allows to rigorously handle partial and noisy observations.

▶ Typical machine learning cost function with $H_k \equiv \mathbf{I}_k$ in the limit $\mathbf{R}_k \longrightarrow \mathbf{0}$:

$$\mathcal{J}(\mathbf{p}) \approx \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{y}_k - M_k(\mathbf{p}, \mathbf{y}_{k-1})\|^2_{\mathbf{Q}_k^{-1}} - \ln p(\mathbf{y}_0, \mathbf{p}).$$

Figure 7.3: Towards hybrid DA / ML methods... Image source: [8]

Chapter 8

DA by PINNs

DA and ML share similarities: they both enable the solution of inverse problems such as parameter identification and model calibration from data.
This is particularly true between Variational Data Assimilation (VDA) and Artificial Neural Networks (ANN) since both imply to solve large dimensional optimization problems of the form: $\min_{param} J(param)$.

The function J is called the cost in VDA and the loss in ML.

Both method classes employ the gradient descent algorithm to minimize J. The adjoint method used to compute the gradient in VDA is equivalent to the back-propagation algorithm employed in ANNs.
In this chapter, we highlight the similarities between VDA and ANNs when used to solve identification problems from observations or model output data. Moreover we present ANN architectures enabling to perform "Physics Informed" machine learning.

8.1 Artificial Neural Networks (ANNs)

8.1.1 Introduction

A bit of history ANNs date back to the 1940s, initially inspired by the structure of the human brain. In the 1960s and 1970s, the development of the Perceptron algorithm and the backpropagation algorithm marked significant progress. However, in the 1980s, limitations were discovered, leading to a period known as the "AI Winter." The resurgence of interest in ANNs came in the 2000s, driven by advances in computing power and the availability of large-scale datasets. This resurgence, combined with novel architectures like Convolutional Neural Networks (CNNs) for image recognition and Recurrent Neural Networks (RNNs) for sequential data, led to breakthroughs in various fields, including computer vision, natural language processing, and reinforcement learning. Today, ANNs are a cornerstone of modern machine learning and find applications in all scientific areas inclmudeing modelling real-world physical phenomena.

The here considered ANNs are simply fully connected Neural Networks. They are also referred to as Feed-Forward Neural Networks or multi-layer perceptrons.
ANNs are defined by their architecture (number of internal layers, number of perceptrons per layer) and their activation functions (e.g., the differentiable tanh and sigmoid functions or the ReLU - rectified linear unit - function).
After training (this will be clarified later), ANNs can represent multi-scale and non-linear features of underlying, unknown operators that represent the data.

ANNs, along with their various versions (CNN, LSTM, Generative NN, etc.), enable remarkably accurate identification of nonlinear trends in data, provided they are trained on a sufficiently large dataset. It's worth mentioning that *predictions may need to be made sufficiently close to the learned data.*

In essence, well-trained ANNs can be seen as highly effective multi-scale interpolators.
Additionally, supervised learning applied to physically-based data provides an alternative approach for building effective models and solving inverse/identification problems.

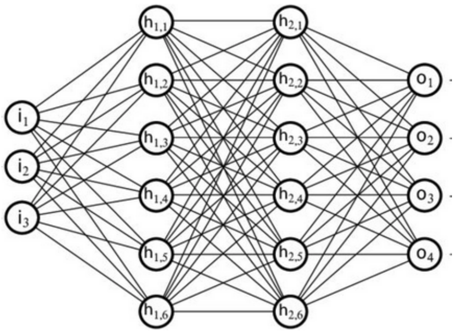


Figure 8.1: A small ANN with 2 hidden layer ($L = 2$). For each connection correspond a weight parameter value and a bias value. Image extracted from [.].

8.1.2 ANNs structure

Let us briefly describe how an ANN can be built up and trained from (large) datasets. The first crucial step is to hold a large training (reliable) dataset describing the targeted phenomena.

Let us consider a dataset $\mathcal{D}$ containing the pairs (X_s^{obs}, Y_s^{obs}), $s = 1, \cdots, N_s$, (called examples or samples in the ML jargon) with X_s the s-th input variable (called 'feature' in the ML jargon) and Y_s the corresponding output (called 'label' in the ML jargon).

Let us denote by $\mathcal{N}_\theta$ an ANN composed of L hidden layers with $\theta = (W, b) \in \mathbb{R}^{N_\theta}$ as its parameters (W the set of weight matrices, b the set of bias vectors). We denote by N_l the neurones number in the l-th layer. $l = 0$ denotes the input layer and $(L+1)$ the output layer. The input and output layers generally have only a few perceptrons.

Let us denote by f_l the l-th layer function, $f_l : x_{l-1} \in \mathbb{R}^{N_{l-1}} \to x_l \in \mathbb{R}^{N_l}$. An ANN can be read as the composition of the $(L+1)$ elementary vectorial functions f_l as follows.
For x an input in $\mathbb{R}^n$ and y the output, $y \in \mathbb{R}^p$,

$$\boxed{\begin{array}{l}\mathcal{N}_\theta : x \in \mathbb{R}^n \mapsto y(\theta)(x) \in \mathbb{R}^p \\ \text{ with } \mathcal{N}_\theta(x) = (f_{L+1} \circ \ldots \circ f_1)(x)\end{array}} \tag{8.1}$$

Given the l-th layer function f_l, we denote by σ_l its activation function and by $\theta_l = (W_l, b_l)$ its parameters.
We have $\sigma_l : \mathbb{R}^{N_l} \to \mathbb{R}^{N_l}$, the weight matrix $W_l^{(N_{l-1},N_l)} \in \mathcal{M}_{N_l,N_{l-1}}(\mathbb{R})$ and the bias vector $b_l \in \mathbb{R}^{N_l}$.
Thus,

$$x_l = f_l(x_{l-1}) = \sigma_l\left(W_l^{(N_{l-1},N_l)} \cdot x_{l-1} + b_l\right), \quad x_l \in \mathbb{R}^{N_l} \tag{8.2}$$

The dimension input variable space n may be small or large, therefore defining a small or large dimensional problem.
The output field y is often large dimensional, particularly in regression problems. However, the final output often measures a quantity through its norm, therefore "gathering" the (numerous) output components onto a one-dimensional quantity only.

8.1.3 ANNs training: the optimization problem

Let us consider a dataset $\mathcal{D}$ containing data pairs (X_s^{obs}, Y_s^{obs}), $s = 1, \cdots, N_s$.
A misfit functional measuring (here simply in 2-norm) the discrepancy between the NN output and the given target value Y^{obs} is defined as:

$$J_{obs}(\mathcal{D}) = \|Y^{obs} - \mathcal{N}_\theta(X^{obs})\|_{2,N_s}^2 = \frac{1}{N_s}\sum_{s=1}^{N_s}\left(Y_s^{obs} - \mathcal{N}_\theta(X_s^{obs})\right)^2 \tag{8.3}$$

Training an ANN consists to minimize this misfit functional $J(\mathcal{D})_{obs}$ *with respect to the NN parameters* θ. Then we define the functional to be minimized as:

$$j_{\mathcal{D},obs}(\theta) = J_{obs}(\mathcal{D}) \tag{8.4}$$

Thus, training the ANN consists to solve the following differentiable optimization problem:

$$\theta^* = \min_\theta j_{\mathcal{D},obs}(\theta) \tag{8.5}$$

For deep ANNs, θ is extremely high dimensional e.g. $\mathcal{O}(10^q)$ with $q = 6$ and more.
Problem (8.5) is therefore a very large dimensional optimization problem requiring advanced (and still under investigations) efficient minimization algorithms, see next section for a few details.

After training, the ANN $\mathcal{N}_\theta$, determined by its architecture and its optimized parameters $\theta^* = (\theta_0^*, \cdots, \theta_{L+1}^*)$, is supposed to accurately represent the underlying unknown operator mapping the input data onto the output data, that is:

$$\mathcal{N}_{\theta^*} \approx \mathcal{F} \text{ where } \mathcal{F} : X^{obs} \mapsto Y^{obs} \tag{8.6}$$

It is an approximation: even after training we do *not* have: $\mathcal{N}_{\theta^*}(X^{obs}) \approx Y^{obs}$ only.

8.1.4 Trained ANNs: surrogate models

After training, ANNs enable to remarkably find nonlinear trends between data therefore providing excellent estimators. If sufficiently well trained, $\mathcal{N}_{\theta^*}$ is supposed to approximate the underlying unknown operator $\mathcal{F} : X^{obs} \mapsto Y^{obs}$, even if the latter is multi-scale and highly non linear.
However, to be "accurate", the predictions given new input data may need to be sufficiently close to the learned data.

Moreover, given a new value of the input parameter, a forward run of an ANN is extremely fast. indeed, it is a simple evaluation of a composition of (numerous) basic functions.
Thus in a modeling context, a (well) trained ANN constitues a *surrogate model.*

Then, if defined from the parameter space $\mathcal{U}$ onto a physics-based model output space $\mathcal{Y}$, or observations space $\mathcal{Z}$, $\mathcal{N}_{\theta^*}(u)$ constitutes a *surrogate of the (corresponding) direct model.*

Conversely, if defined from the state space $\mathcal{Y}$, or observations space $\mathcal{Z}$, onto the parameter space $\mathcal{U}$, $\mathcal{N}_{\theta^*}(u)$ constitutes a *surrogate of the inverse model.*
Moreover, nowadays, a few progresses remain to be done in particular for large dimensional problems, see e.g. [16] for a review. Also the resulting surrogate model may be not as accurate as good physical-based models (when existing).

Case of an output functional

In a modeling context, the user is often interested in a particular quantity of interest $g(y)$, $g : \mathbb{R}^p \to \mathbb{R}$, with y deriving from the ANN output, $y = \mathcal{N}_\theta(x)$.
Let us cite an example: $y(x)$ be a temperature depending on a material property x, $g(y)$ the total energy in a given subdomain.
Then, by introducing the following operator $\mathcal{J}_\theta$,

$$\mathcal{J}_\theta(x) = g \circ \mathcal{N}_\theta(x), \quad \mathcal{J}_\theta : \mathbb{R}^n \to \mathbb{R} \tag{8.7}$$

Learning this functional $\mathcal{J}_\theta$ consists to solve the following optimization problem:

$$\theta^* = \underset{\theta}{argmin} \left(\|g(Y^{obs}) - \mathcal{J}_\theta(X^{obs})\|^2_{2,N_s} \right) \tag{8.8}$$

by employing the same strategies as for $\mathcal{N}_\theta(x)$.

8.1.5 Optimization strategies and ANNs internal technics

Optimization strategies

The choice activation function σ of the perceptrons depend on the required properties:
- the sigmoid or tanh function provide a differentiable NN (therefore's omehow regular, smooth),
- the rectified linear unit (ReLU) function is the prefered choice for information reduction or extraction.

To numerically minimize $j_{\mathcal{D}}(\theta)$, first-order gradient-based optimization algorithms are employed.
For relatively small dimensional problem up to large dimensional problem (say up to $\dim(x) = O(10^6)$), the deterministic L-BFGS descent algorithm is a good choice as it is quite accurate.

For larger dimensional problems, a stochastic descent algorithm is required: the ADAM algorithm [28] is nowadays considered the reference algorithm.

Note that for moderately large problems (say $dim(x) \approx O(10^6)$), a good strategy consists to first performing the ADAM algorithm, and then applying the L-BFGS algorithm. This enables a good balance between convergence speed and final accuracy.

ANNs hyper parameters

The so-called hyper-parameters of the NN are the learning rate, decay rate, dropout probability, see e.g. [?] for details. These so-called ”hyper-parameters” are mainly experimentally chosen. The selected values are those providing the minimal value of $j_{\mathcal{D}}(\cdot)$.
In a modelling context, these hyper-parameters can be seen as ”priors” of the model and it can be an issue to be tune them (similarly to a regularization term weight α in bi-objective optimization).

Gradient computation by automatic differentiation

First-order descent algorithms require the computation of the large dimensional cost gradient $\nabla J(\theta)$. Computing efficiently such large dimensional gradients, $\dim(\nabla J(\theta) \sim \dim\,(\theta) \approx O(10^6 - 10^9)$, may be a challenge. However, ANNs presents a very simple particular form: it is a composition (of numerous however) elementary functions, see (8.1).

As a consequence, the cost function gradient $\nabla J(\theta)$ can be computed by applying the differential chain rule to this composition form.
This is what is done by the so-called "back-propagation" procedure. This step is performed using Automatic Differentiation. The basic principles of Automatic Differentiation can be found in Section **??**; also for details the reader may refer e.g. to [6] and references therein.

Nowadays, ANNs can be easily coded in Python using one of the libraries available online such as PyTorch - Mpi4Py, TensorFlow, Scikit-NN etc.

8.2 ANNs to solve u-parametrized equations

This section has been conceptualised with Hugo Boulenc during his PhD investigations (INSA-IMT, 2022-25). The figures have been produced by H. Boulenc.

Let us consider the same general u-parametrised PDE-based model as in (9.2):

$$A(u;y)(x) = B(u)(x) \text{ for } x \in \Omega \tag{8.9}$$

with Ω an open subset of $\mathbb{R}^d$.
The physical-based model $A(u;y)$ is a-priori non-linear both in u and y.

8.2.1 Fully-parametrized ANN

Let us consider the pair $(x;u(x))$ as the input variables of the ANN: x the space variable, $u(x)$ the PDE parameter.

Given an output quantity of interest $g(y(x))$, $y(x)$ the model output (= the state of the system), the ANN can be schematized as on Fig. 8.2.

After learning, the trained ANN $\mathcal{N}_{\theta^*}$ is expected to be an approximation of the model operator $\mathcal{M}$ defined as:

$$\mathcal{N}_{\theta^*} \approx \mathcal{M} \text{ with } \mathcal{M} : (u;x) \mapsto y(u;x) \tag{8.10}$$

with $y(u;x)$ satisfying the u-parametrized model equation (8.9).

Again the question of dimension of the inverse problem is determining.

- In the case u is large dimensional, the training is hardly feasible due to the extremely numerous samples required.
- In the case u is not high large dimensional, after learning, one may perform $\mathcal{N}_{\theta^*}(u;x)$ given non-learned values of $(x;u(x))$ as predictor.
  Moreover, in the case u is very low dimensional ($\dim(u) = \mathcal{O}(1)$), one can infer values of

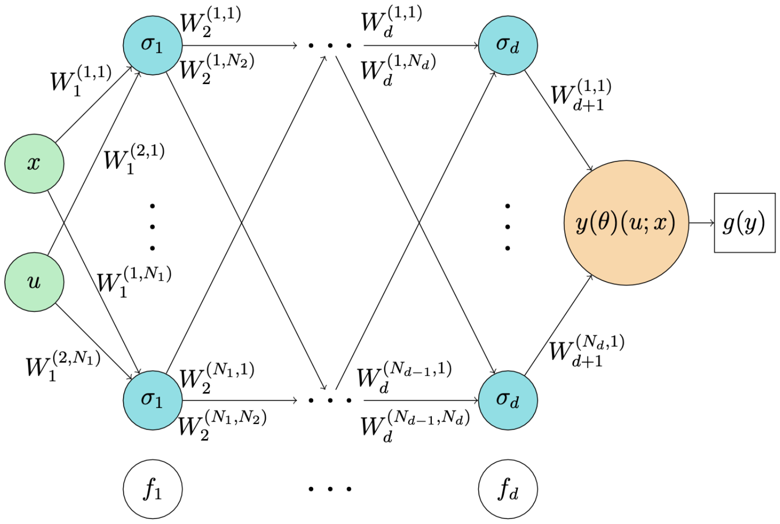


Figure 8.2: ANN to approximate the output of a u-parametrized model : the fully-parametrized ANN version.

(For a sake of clarity, the biases b_l are not indicated on the figures).

u given data $y^{obs}(x)$ using standard optimization routine(s).

An example for a simple model with low dimensional parameter u is presented in Section 8.4.

8.2.2 Semi-parametrized ANN

For moderate dimensional parameter u, say $\dim(u) = O(10^q) \quad q \approx 2$, a the so-called semi-parametrized version of NN is a good option. This consists to build an ANN as indicated on Fig. 8.3. An example for a simple model with quite low dimensional parameter u is presented in Section 8.4.

On the gradient computations

As already mentioned, a key point of the efficiency of ANNs is the possibility to easily compute the output functional gradient with respect to the input parameter. This is done by applying the chain rule to the large composition of the elementary functions $f_l(x)$, see (8.1) by automatic algorithmic (Automatic Differentiation process).

Obviously, the computation complexity (in terms of operations numbers therefore CPU-time) is very different if considering large dimensional problem (with a large dimension of the model parameter) or small dimensional ones. Moreover, the gradient expression is not the same for the fully-parametrized NN than for the semi-parametrized one. The mathematical expression of the gradients are detailed later in both cases in Section .

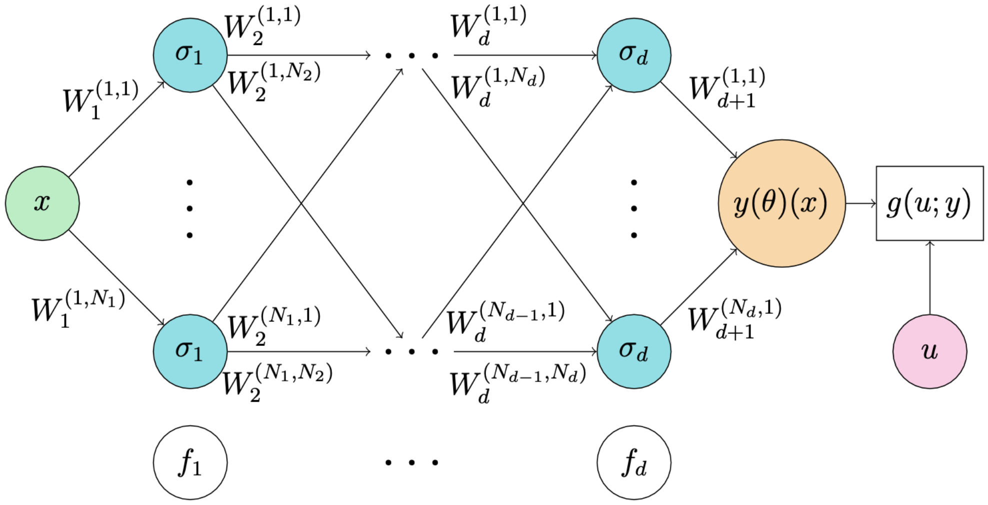


Figure 8.3: ANN to approximate a parametrized PDE-based model : the semi-parametrized version.

8.3 Physics-Informed Neural Networks (PINNs)

An important drawback of ANNs is their lack of explainability and reliability.
PINNs consist to minimize both the misfit to the observations (the same loss function J_{obs} as for standard ANN) plus an additional term: the residual of the underlying physical model (term denoted by J_{res}).
In context of modeling physical phenomena, a way to address this issue consists to introduce the physical equation as a constraint in the minimization procedure. This can be simply done by adding the residual of the model into the loss function. This principle is those of the so-called PINNs introduced in [30, 37].

8.3.1 Basic formalism

The residual of the model (**??**) reads as:

$$r(u;y) = A(u;y) - L(u) \tag{8.11}$$

The corresponding residual functional $J_{res}(u;y)$ is defined by:

$$J_{res}(u;y) = \|r(u;y)\|_{2,\mathcal{X}_{col}} \tag{8.12}$$

where $\mathcal{X}_{col}$ denotes a set of points within the domain Ω. It is may be perceived as collocation points where the residual is evaluated.

The total objective function J, $J : \mathcal{U} \times \mathcal{Y} \to \mathbb{R}$, is then defined by:

$$J_\alpha(u;y) = J_{\mathcal{D},obs}(y) + \alpha J_{res}(u;y) \tag{8.13}$$

with J_{obs} the standard misfit term defined by (8.3).

Training a PINNs as proposed in [37] consists to solve the optimization problem (8.5), that is

$$\theta^* = \min_\theta j_{\mathcal{D},obs}(\theta), \tag{8.14}$$

with the loss function defined by

$$j_{\mathcal{D},obs}(\theta) = J_\alpha(u; y(\theta)), \tag{8.15}$$

$J_\alpha(u; y)$ defined by (8.13), and with a semi-parametrized architecture as those indicated on Fig. 8.4.
The same minimization strategies as for standard NNs $\mathcal{N}_\theta(x)$ are employed.

PINNs rely on Automatic Differentiation Finally, the most important remark relies on the way to compute the residual $r(u; y) = A(u; y) - L(u)$.
Residual of PDEs involves partial derivatives as $\partial_{x_1} y(x), \partial^2_{x_2 x_2} y(x)$ etc, depending on the PDE order and expression. The space variable $x = (x_1, \ldots, x_d)$ is an input of the ANN. Therefore Automatic Differentiation of the ANN can provide any partial derivative $\partial^q_{x_j \ldots x_l} y$ therefore simply evaluating the residual (8.11). This is very likely the most important trick of the PINNs concept, [37, 34].

Minimizing the residual vs minimizing the error

Let us recall that PINNs consist to minimize both the misfit to the observations (the term J_{obs}) and the residual (the term J_{res}). In this paragraph, we wonder if the minimization of the residual implies the minimization of the error.

Given u, we have an unique (exact) solution y^* satisfying: $r(u; y^*) = A(u; y^*) - L(u) = 0$.
Given an approximation $\tilde{y}$ of y^*, we have of course: $r(u; \tilde{y}) = A(u; \tilde{y}) - L(u) \neq 0$.
We define the error $\varepsilon(y)$ by $\varepsilon(y) = (y^* - \tilde{y})$.

Let us assume that $A(\cdot; y)$ is linear in y. In this case, we have:

$$r(u; \tilde{y}) = A(u)\tilde{y} - A(u)\tilde{y} = A(u)\varepsilon(y)$$

Therefore

$$\begin{array}{rcl} \varepsilon(y) & = & A^{-1}(u)\; r(u; \tilde{y}) \\ \|\varepsilon(y)\|_2 & \leq & \|A^{-1}(u)\|_2 \|r(u; \tilde{y}))\|_2 \end{array}$$

This estimation shows that if the residual vanishes then the error vanishes too. However, a small residual value does not necessarily implies a "small" error value... Indeed, this depends on the spectrum of A^{-1}, equivalently on the spectrum of A.

In finite dimension, we have, see e.g. [?]: $\|A^{-1}(u)\|_2 \leq \min_i |\lambda_i(A^{-1})|$. Therefore the estimation:

$$\|\varepsilon(y)\|_2 \leq \frac{1}{\max_i |\lambda_i(A)|} \|r(u;\tilde{y}))\|_2 \tag{8.16}$$

A small residual implies a small error on the state if $\max_i |\lambda_i(A)|$ "large enough".

8.3.2 PINNs for direct modeling

Here, we seek to solve the model equation $A(y) = L$ in Ω by employing a PINNs.
We then consider the architecture indicated on Fig. 8.4 which does not consider any parameter as input variable.

After training, the PINNs is supposed to provide a surrogate model, such that:

$$\mathcal{N}_{\theta^*}(x) \equiv \tilde{y}(\theta^*)(x) \approx y^*(x) \tag{8.17}$$

with $y^*(x)$ solution of the model.

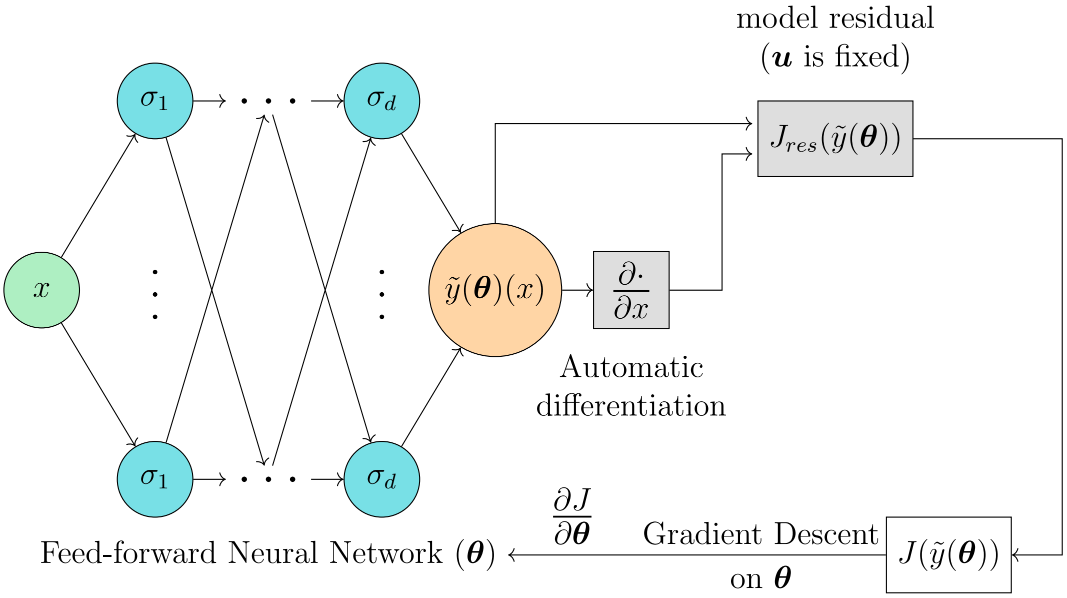

model residual
($\boldsymbol{u}$ is fixed)
σ_1
σ_d
σ_1
σ_d
x
$J_{res}(\tilde{y}(\boldsymbol{\theta}))$
$\tilde{y}(\boldsymbol{\theta})(x)$
$\frac{\partial \cdot}{\partial x}$
Automatic differentiation
Feed-forward Neural Network ($\boldsymbol{\theta}$)
$\frac{\partial J}{\partial \boldsymbol{\theta}}$
Gradient Descent on $\boldsymbol{\theta}$
$J(\tilde{y}(\boldsymbol{\theta}))$

Figure 8.4: PINNs-like architecture to build a surrogate (non-parametrized) direct model. After training, the NN output approximates the observations while minimizing the residual of a physical model.

8.3.3 PINNs for inverse modeling

Here, we seek to solve the u-parametrized equation $A(u; y) = L$ in Ω by employing a PINNs, and to estimate the parameter u^* corresponding to the given observations set.

To do so, we build the architecture indicated on Fig. 8.5 which does not consider any parameter as input variable, however u appears in the term J_{res}.

After training, the PINN provides a surrogate model, such that:

$$\mathcal{N}_{\theta^*}(x) \approx y^*(u^*)(x) \tag{8.18}$$

with $(u^*, y^*)(x)$ a solution minimizing the loss function defined by (8.15).

Remark. In both cases, direct and inverse model surrogates, the loss gradient is computed by AD. We refer e.g. to [34] and references therein for more details on PINNs.
Note that the mathematical expressions of the loss gradient can be calculated. This is what is done in a next section. These "gradient" expressions enables a better understanding of the ANN features.

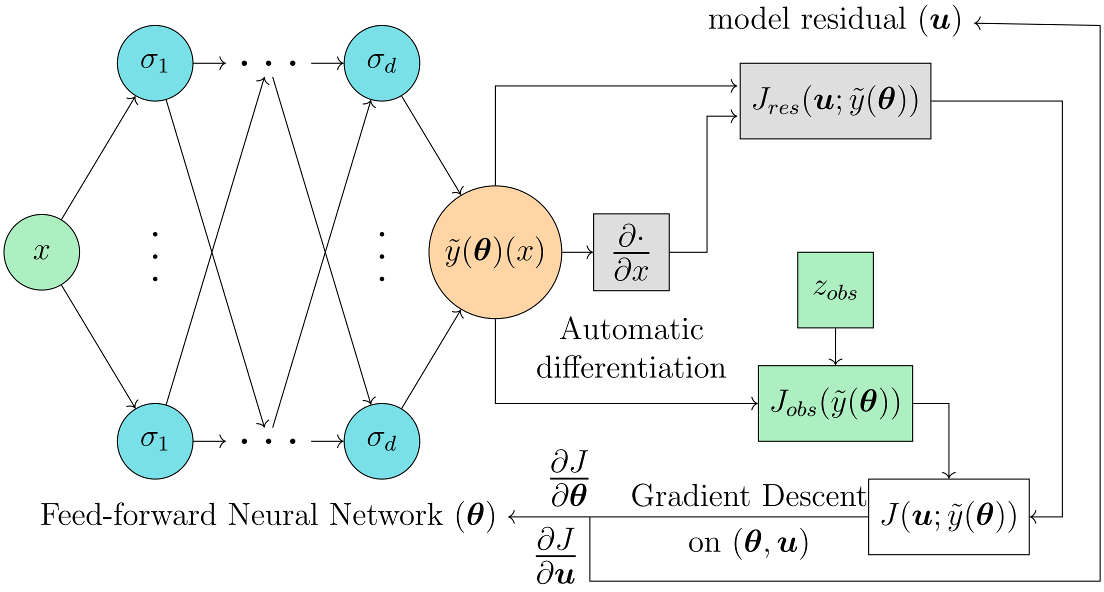

model residual $(\boldsymbol{u})$
σ_1
σ_d
$J_{res}(\boldsymbol{u}; \tilde{y}(\boldsymbol{\theta}))$
x
$\tilde{y}(\boldsymbol{\theta})(x)$
$\frac{\partial \cdot}{\partial x}$
z_{obs}
Automatic differentiation
$J_{obs}(\tilde{y}(\boldsymbol{\theta}))$
σ_1
σ_d
$\frac{\partial J}{\partial \boldsymbol{\theta}}$
Gradient Descent on $(\boldsymbol{\theta}, \boldsymbol{u})$
$J(\boldsymbol{u}; \tilde{y}(\boldsymbol{\theta}))$
Feed-forward Neural Network $(\boldsymbol{\theta})$
$\frac{\partial J}{\partial \boldsymbol{u}}$

Figure 8.5: PINNs-like architecture to solve a parameter identification problem. After training, both u^* and $y(u^*)$ are estimated.

8.4 Examples

A reference library addressing PINNs-like architecture is the DeepXDE library [1]. Various examples of direct and inverse problems solved by PINNs-like architectures are proposed.

For examples locally developed at IMT-INSA Toulouse, please consult the supplementary material.

[1] https://deepxde.readthedocs.io/en/latest

Part III

Variational Approaches for ODEs and General PDEs-based models

Chapter 9

Variational DA in simple finite-dimensional cases

The outline of this chapter is as follows.

Contents

9.1 The inverse problem

The previous estimation problems aimed at estimating u satisfying the linear equation $\mathcal{M}(u) = z_{obs} + \varepsilon$, with $u \in \mathbb{R}^n$, $z_{obs} \in \mathbb{R}^m$, with M observations given, $z_{obs} = (z_{obs,1}, \dots, z_{obs,M})$.
The VDA approach consists to minimize the cost function $j(u)$ defined as in (6.17).

In the LQG case (in particular $\mathcal{M}$ is a linear operator), $u^* = arg\min_u j(u)$ is the same solution as the BLUE, see Prop. 5.2, and the MAP, see (6.16).

The variational approach is primarily targeted at large dimensional and/or non-linear problems since the formulation naturally extends to such contexts. Indeed, the resulting VDA algorithms (3D-Var, 4D-Var and so on, as detailed later) remain performant for large dimensional non linear problems.

In what follows, the inverse problem is as follows.

Estimate $u(x)$ such that:
$A(u(x); y(x)) = B(u(x))$ in Ω accompan
given observations of the system: $z_{obs}($

$A(u; y)$ denotes a PDE-based operator.

Examples
- The Electrical Impedance Tomography (EIT) problem presented in the introduction, see Section 1.2.

- The physical-based model is defined by:

$$A(u; y) = -div(\lambda(u)\nabla y) + y^3$$

with $\lambda(\cdot)$ a given function, and the RHS $B(u) = g(\|\nabla u\|)$ with $g(\cdot)$ a given function.
The observation operator is defined by $z_{obs} = \partial_n y$ on a piece of the boundary Γ, $\Gamma \subset \partial\Omega$.

Remark Contrary to the classical "explicit" least square problems as (2.1), the unknown u is related to the measurements z_{obs} through the parameter-to-state operator $\mathcal{M}$, $\mathcal{M} : u \mapsto y(u)$, $y(u)$ solution of (9.1) with u given.
As a consequence, even in the linear case ($\mathcal{M}(u)$ linear), the optimal solution $u^* = arg\min_u j(u)$ cannot be obtained by simply solving the normal equations (2.3).

VDA aims at estimating $u(x)$ by employing an optimal control approach of the physically-based model.

However *in the particular case where the parameter u is the RHS of the equation*, the optimal estimate u^* can be derived without introducing optimal control concepts. This is what is done in a next section: we mathematically solve by VDA the identification problem for linear operators $A(u; \cdot)$ where the parameter u equal the RHS ($B(u) = u$).

This allows us to illustrate the basic principle of VDA methods while employing only simple algebraic tools.

The general case which requires more complex mathematical tools and concepts is addressed in Part II of this manuscript, see e.g. Chapter 11, after a presentation of the optimal control methods for ODEs (Chapter 10).

9.2 The VDA formulation

We here present a VDA formulation for a general non linear stationary PDE-based model.

9.2.1 The direct model and the parameter-to-state operator $\mathcal{M}$

The considered PDE-based model reads as follows.Given an unknown/uncertain parameter $u(x)$, find $y(x)$ which satisfies the Boundary Value Problem (BVP):

$$A(u(x)); y(x)) = B(u(x)) \text{ with } x \in \Omega \subset \mathbb{R}^d \tag{9.2}$$

accompanied by Boundary Conditions (BC).

The PDE-based operator $A(\cdot;\cdot)$ is a-priori non-linear both in u and y that is the maps $u \mapsto A(u;\cdot)$ and $y \mapsto A(\cdot;y)$ are non-linear.

Given the parameter u, the direct model (9.2) is supposed to have a unique solution y which is then denoted by y^u or $y(u)$.

The parameter-to-state map (also previously called the model operator) $\mathcal{M}$ is defined as:

$$\mathcal{M} : u(x) \mapsto y(u)(x)$$

with $y(u)$ solution of (9.2) given u.

9.2.2 The observation operator and the cost function

Data (also called observations or measurements) are denoted by z^{obs}. Data are not necessarily of same nature than the state of the system y.

For example, z^{obs} denotes measured wavelength electromagnetism signals which have to be next compared to a temperature y, the model output (state of the system).
We introduce the *observation operator* Z which maps the state of the system y onto the observations space as:

$$Z : y(x) \in \mathcal{Y} \mapsto Z(y(x)) \in \mathcal{Z} \tag{9.3}$$

The observation operator $Z(\cdot)$ can be a linear or not.
In real-world problems, Z can be more or less complex e.g. a multi-scale non linear model or

simply the Identity if measuring directly the state of the system.
Given observations z^{obs}, one naturally wants to minimize the misfit term $\|Z(y(u)(x))-z^{obs}(x)\|^2_{R^{-1}}$. As already mentioned, in real-world problems and because of lack of information, the observation norm R^{-1} is often assumed to be diagonal: $R^{-1} = diag(\rho_{obs,1}, \dots, \rho_{obs,m})$. Then, the coefficients $\rho_{obs,m}$ simply represent the confidence one has on each observation.

Moreover, as discussed in Section 2.2, it is often necessary to introduce a regularization term in the minimized cost function.
We introduce the objective function $J(u; y)$ defined as:

$$\boxed{J(u;y) \;=\; J_{obs}(y) + \alpha_{reg} J_{reg}(u)} \tag{9.4}$$

with

$$J_{obs}(y) = \|Z(y) - z^{obs}\|^2_{R^{-1}} \text{ and potentially } J_{reg}(u) = \|u - u_b\|^2_{B^{-1}} \tag{9.5}$$

Then the cost function $j(u)$ is defined from the observation function $J(u; y^u)$ as:

$$\boxed{j(u) = J(u\,; y^u) \text{ where } y^u \text{ is solution of the model (9.8).}} \tag{9.6}$$

The cost function $j(u)$ depends on the control u through the state of the system $y^u = \mathcal{M}(u)$. The term $J_{reg}(u)$ is a regularization Tykhonov-type term, see Section **??** for further discussions. The present expression penalizes the discrepancy with the prior u_b in norm B^{-1} whose the definition B^{-1} can be important too. [1]. It has been shown in the previous chapters that in the LQ case, the optimal estimation relies on the covariance errors matrices.

u_b is "background" value of u therefore a prior of the inverse problem. Other regularization terms such as e.g. $\|\nabla u\|^2_2$ can be considered.

The scalar parameter α_{reg} is the weight parameter to balance the two terms. Its definition may be tricky. We refer the reader to Chapter 1 for these concepts.

9.2.3 The optimization problem

Solving the VDA problem consists to find:

$$\boxed{u^*(x) = arg \min_{u(x)\in\mathcal{U}} j(u(x))} \tag{9.7}$$

This is an optimal control problem.
The solution u^* (if existing) is an optimal control of the system. This is the estimate we are looking for. This is the VDA solution.
In others words, the VDA approach consists to solve a *non-linear least-square problem with an underlying physical model.* The latter constitutes a constraint of the optimization problem.

[1] The notations R^{-1} and B^{-1} are the classical notations in the DA community, [24].

9.3 Linear model, finite dimensional case

We now derive the VDA solution in the case *the model is linear and finite dimensional.*
We develop here the calculations in finite dimension since more simple than in infinite dimensions. Indeed, in infinite dimensions, e.g. in Banach and Hilbert spaces, the calculations require some extra knowledge in differential calculus, in particular for non-linear problems. Calculations in infinite dimensions for general non linear problems are addressed in Part II.

9.3.1 Problem statement

Let u be the control variable, $u \in \mathbb{R}^m$. Let the model be linear and represented by $A(u)$, $A(u) \in \mathcal{M}_{n\times n}$ a non singular real matrix. The (direct) model reads:

$$\boxed{\left\{ \begin{array}{l} \text{Given } u \in \mathbb{R}^m, \text{ find } y \in \mathbb{R}^n \text{ such that:} \\ A(u)\, y \;=\; F(u) \end{array} \right.} \tag{9.8}$$

with the RHS $F(u) \in \mathbb{R}^m$.

Given u, the solution of the direct model is denoted by y^u. It is the *state of the system.*

Let $J(u;y)$ be the objective function measuring the misfit between the state y and the observations defined as in (9.4) (9.5).
However, here the observation operator Z is linear: $Z \in \mathcal{M}_{n\times n}$. Therefore:

$$\boxed{J(u;y) = \|Zy - z^{obs}\|^2_{R^{-1}} + \alpha_{reg}\|u - u_b\|^2_{B^{-1}}} \tag{9.9}$$

with R^{-1} and B^{-1} symmetric positive semi-definite matrices of dimension $n \times n$.

Recall that the cost function $j(u)$ to be minimized is defined from the observation function J as: $j(u) = J(u\,;y^u)$ with $y^u \equiv y(u)$ is solution of (9.8).

Recall that the cost function $j(u)$ is defined as $\boxed{j(u) = J(u;y^u)}$, y^u the unique solution of the direct model.

The estimation problem consists to find $u^* \in \mathbb{R}^m$ such that: $\boxed{j(u^*) = \min_{U_h} j(u)}$.

The set U_h denotes here a subset of $\mathbb{R}^m$ which may include inequality constraints or equality constraints on u (thus defining an optimization problem with additional constraints).

9.3.2 On the numerical resolution of the VDA problem

The VDA problem consists to solve the optimization problem $arg\min_{U_h} j(u)$ above. Few approaches are a-priori possible: roughly, global optimization methods or local minimization methods. The choice mainly depends on the CPU time required to evaluate the cost function $j(u)$ and on the control variable dimension m, therefore the dimension of the gradient $\nabla j(u)$.

- **Low complexity problems**. If the CPU time required to evaluate the cost function $j(u)$ is negligible (let say in fractions of seconds using your laptop or a super-computer, whatever), then one can adopt a *global optimization approach* based on stochastic algorithms e.g. Monte-Carlo type algorithms, heuristic methods (e.g. genetic algorithms) or surface response approximation.
- **CPU-time consuming problems or large dimensional parameters** u. On contrary, if the dimension of the parameter u is large ($m = dim(U_h) = O(10)$ and more), moreover if the computation of the cost function $j(u)$ is CPU-time consuming (this the case e.g. if considering a 3d PDE model), then global optimization is not worth considering. Then, one has to adopt *local minimization* approaches based on *algorithms of descent*.
  Then the computation of the cost function gradient $\nabla j(u)$ is required, see any good book or course on numerical optimization, see also Appendix.
  The computation of the gradient $\nabla j(u)$ in the large dimensional case ($m >>$) is tricky. This is the point presented in the present simple case in next section.

9.3.3 Computing the cost function gradient $\nabla j(u)$

A brute force approach: approximation by Finite Differences

Given $u \in \mathbb{R}^m$, a basic approach consists to *approximate the gradient* using Finite Differences (FD) such as:

$$\nabla j(u) \cdot \delta u \approx \frac{j(u + \varepsilon \delta u) - j(u)}{\varepsilon} \tag{9.10}$$

with δu a given direction, $\delta u \in \mathbb{R}^m$.

Pros and cons of this brute force approach are as follows. ⊕ This is a simple way to compute the gradient.

⊖ This numerical approach requires $(m + 1)$ evaluations of $j(u)$ therefore $(m + 1)$ resolutions of the direct model. This is generally not possible for m large!...
⊖ The obtained accuracy of $\nabla j(u)$ is not controlled since depending on an the choice of ε. An optimal value of ε is unknown.
And, In large dimension, the descent algorithm is generally sensitive to the gradient accuracy.

The few differentials

Recall that $j : \mathbb{R}^m \to \mathbb{R}$, $j(u) = J(u; y(u))$, with $J(u; y)$ defined by (9.5).

We have the gradients $\nabla j(u) \in \mathbb{R}^m$, $\nabla_u J(u;y) \in \mathbb{R}^m$ and $\nabla_y J(u;y) \in \mathbb{R}^n$.

The differential $D_u y(u)$ is here a $n \times m$-Jacobian matrix.

$D_u y(u)$ represents the derivative of the state $y(u)$ with respect to the parameter u.

Example. $D_u y(u)$ represents the derivative of a temperature field $y(x)$ with respect to the spatially-distributed source term $u(x)$.

A first expression of $\nabla j(u)$

Formally, the state of the system satisfies the relation: $y(u) = (A(u))^{-1} F(u)$.
Of course, in practice, one never compute A^{-1}, instead the linear system is solved by a linear algebra method (e.g. a Gauss-type algorithm).
However, the cost $j(u)$ reads explicitly in function of u as:

$$j(u) = J(u; y(u)) = \|Z(A(u))^{-1}F(u) - z^{obs}\|^2_{R^{-1}} + \alpha_{reg}\|u - u_b\|^2_{B^{-1}} \tag{9.11}$$

From the relation $j(u) = J(u; y(u))$, we get for all $v \in \mathbb{R}^m$,

$$\boxed{< \nabla j(u), v > = < \nabla_u J(u; y(u)), v) > + < \nabla_y J(u; y(u)), D_u y(u) \cdot v >} \tag{9.12}$$

where $< \cdot, \cdot >$ denotes here the Euclidian scalar products (either in $\mathbb{R}^m$ or in $\mathbb{R}^n$).

Recall that: $y(u) = (A(u))^{-1}F(u)$. Therefore:

$$\boxed{D_u y(u) = D_u \left((A(u))^{-1}\right) F(u) + (A(u))^{-1} F'(u)} \tag{9.13}$$

The term $D_u((A(u))^{-1})$ represents the differential of the inverse of the direct model operator. This term and the expression (9.13) of $D_u y(u)$ are generally *not* tractable.

In the simple case considered in next section, this difficulty can be easily circumvented.
On the contrary, in general cases, this requires more mathematical developments which are presented in a next chapter.

9.3.4 A simple case: u in the RHS only

For sake of simplicity, let us consider from now a simplified u-parametrized model: the parameter u appears in the RHS of the model only, and not in the differential operator A.
In this case, the inverse problem solution is much easier to characterize. Indeed,in this case,

explicit calculations can be derived and we can easily introduce the concept of adjoint equations.

The considered u-parametrized direct model equation reads:

$$\boxed{Ay = Fu} \tag{9.14}$$

Example. The illustrative problem presented in Section **??** fits into the present framework if considering the following simplified (unphysical) equation:

$$-\Lambda_{ref}(x)\partial^2_{xx}H(x) + \partial_x H(x) \;=\; \partial_x b(x) \text{ in } (0, L) \tag{9.15}$$

with here Λ_{ref} independent of b, $\Lambda_{ref} = \frac{(H_{ref}-b_{ref})}{|\partial_x H_{ref}|}$.

The gradient expression in the simple case

From the direct model expression (9.14), we simply have: $\boxed{D_u y(u) = A^{-1}F}$, expression to be compared to (9.13).

Then, for all $v \in \mathbb{R}^m$,

$$< \nabla j(u), v > = < \nabla_u J(u; y(u)), v) > + < F^T A^{-T} \nabla_y J(u; y(u)), v >$$

From the generic expression (9.4) of $J(u; y)$, we get:

$$\nabla_u J(u; y(u)) = J'_{reg}(u) \text{ and } \nabla_y J(u; y(u)) = J'_{obs}(y)$$

From (9.9), we get:

$$\nabla_u J(u; y(u)) = 2\alpha_{reg} B^{-1}(u - u_b) \text{ and } \nabla_y J(u; y(u)) = 2Z^T R^{-1}(Zy(u) - z^{obs})$$

Therefore the gradient expression:

$$\boxed{\nabla j(u) = \; F^T A^{-T} \nabla_y J(u; y(u)) + \; J'_{reg}(u)} \tag{9.16}$$

with $\nabla_y J(u; y(u)) = J'_{obs}(y) = 2Z^T R^{-1}(Zy(u) - z^{obs})$.

Why an adjoint equation? Because it is not tractable to compute A^{-T}.
Actually, computing the term $\left[A^{-T}\nabla_y J(u; y(u))\right]$ consists to solve the following linear equation:

$$\boxed{A^T p = \nabla_y J(u; y(u))} \tag{9.17}$$

This is the so-called *adjoint equation.* p denotes an additional field, $p \in \mathbb{R}^n$. It is the *adjoint state.*

If A is non-singular therefore its "adjoint" A^T too. Since the unique solution p depends on u, we denote it by $p(u)$ too. Note that p depends on $y(u)$ too.

Then, the gradient expression not explicitly dependent on A^{-T} reads as:

$$\boxed{\nabla j(u) = F^T p(u) + \ J'_{reg}(u)} \tag{9.18}$$

with $p(u)$ the unique solution of the adjoint equation (9.17) given u.

This additional field p enabling to obtain the gradient expression above is called the *adjoint state* of the system. It is by definition the solution of the adjoint equation (9.17).
The adjoint equation is the one which enable to derive an expression of $\nabla j(u)$ not explicitly in function of A^{-T}.

Characterization of the optimal solution u^*

The first order necessary condition of optimality reads: $\nabla j(u) = 0$.
However, the expression of $\nabla j(u)$ depends on $p(u)$ which depends on $y(u)$...

We then have a fully coupled system to solve to obtain the expected solution u^*,
$u^* = arg \min_{\mathcal{U}} j(u)$.

This system, called the *optimality system*, contains the three equations characterizing the optimal solution $(u^*, y(u^*))$: the state equation (direct model), the adjoint equation and the first order optimality condition.

The optimality system reads:

$$\boxed{\left\{\begin{array}{ll} \textit{The direct equation:} & Ay = Fu \\ \textit{The adjoint equation:} & A^T p = J'_{obs}(y) \\ \textit{The first order condition:} & \nabla j(u) = \big(F^T p + \ J'_{reg}(u)\big) = 0 \end{array}\right.} \tag{9.19}$$

with here: $\nabla_y J(u; y) = J'_{obs}(y) = 2Z^T R^{-1}(Zy - z^{obs})$ and $J'_{reg}(u) = 2\alpha_{reg} B^{-1}(u - u_b)$.

Observe that the term measuring the misfit to data is in the RHS of the adjoint equation.

The calculation of the cost function gradient $\nabla j(u)$ can be done for general non-linear models, under assumptions of differentiability of the map $u \mapsto y(u)$ of course.
This calculation is done for general non linear problems in infinite dimensions in a next chapter. Calculations in infinite dimensions require higher mathematical backgrounds, however, they enable to rigorously derive the expressions.

9.4 Algorithms to solve the optimality system

The question is here how to solve the optimality system (9.19) characterizing the optimal solution u^* ?

Recall that we assume to be here in a context where the dimension of the control variable u is large. As a consequence, gradient-based local minimization methods only are affordable (versus e.g. gradient-free global optimization methods).

9.4.1 Newton-like approach

After solving the direct equation and the adjoint equation, using FE or FV schemes for example, the cost function and the cost gradient straightforwardly follows by numerical integration. The finite dimensional gradient value $\nabla j(u_h)$ follows for any $u_h \in U_h$.
Next, to solve the first order optimality condition $\nabla j(u_h) = 0$, a first natural idea consists to employ the Newton-Raphson algorithm based here on the derivative of $\nabla j(u)$, that is the Hessian $H_j(u)$...

For small control dimension ($\dim(U_h) <<$), the computation of $H_j(u)$ may be tractable.

For large control dimension, say $\dim(U_h) = O(10^p)$ with p greater than 2 and more, the computation of $H_j(u)$ is a-priori too much memory and CPU time consuming. In this case, first order descent algorithms based on the gradient information only are preferred. In practice, we use Quasi-Newton methods like the BFGS algorithm or the limited memory version L-BFGS, see e.g. [10]. A few recalls on the Quasi-Newton methods and the BFGS algorithm in particular can be found in Appendix.

9.4.2 The 3D-Var algorithm

We here assume that the control dimension is sufficiently large to enforce to use first order descent algorithms like the L-BFGS algorithm.
The resolution of the optimality system is then iteratively solved: given a *first guess* u_0, the current iterate u_m is computed to decrease the cost function.
The resulting algorithm called 3D-Var is schematized in Fig. (9.1).

Given the current control value u,

1) compute the cost function $j(u)$ from the direct model output y^u,
2) compute the gradient $\nabla j(u)$ from the adjoint model output p^u and the direct model output y^u,
3) given u, $j(u)$ and $\nabla j(u)$, compute the new iterate u^{new} as $u^{new} = u + \alpha d(\nabla j(u))$ where $d \in \mathbb{R}^m$ is the descent direction and $\alpha \in \mathbb{R}^+_*$ the step in the linear search. The descent algorithm simply ensures that:

$$j(u^{new}) < j(u)$$

*) Iterate until convergence.

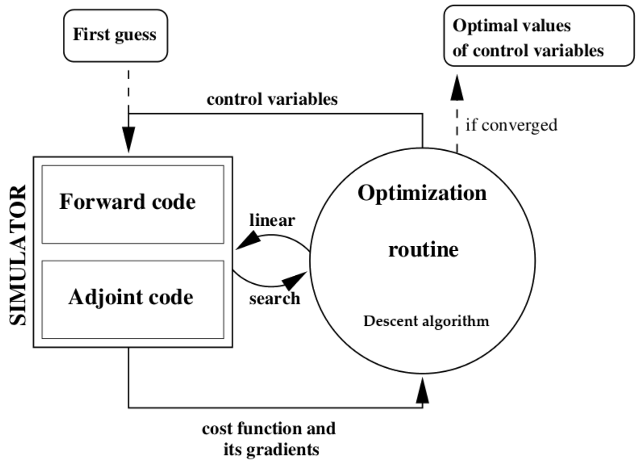


Figure 9.1: VDA algorithm: optimal control of the PDE system. For stationary PDEs, this provides the so-called 3D-Var algorithm, and the so-called 4D-Var algorithm in the unsteady PDEs case (case studied later).

Finally let us recall that if the cost function term $J_{obs}(y)$ presents different local minima or presents ”nearly flat valleys” then the choice of the first guess value, the regularization term J_{reg} and the norms R^{-1}, B^{-1} highly influence the optimal value u^*.

9.5 Example: the illustration problem based on a simplified model

Let us consider a simplified version of the model in the spatial hydrology problem, see Section **??** where the RHS only depends on the control variable $b(x)$.
We consider the following equation:

$$-\Lambda_{ref}(x)\partial^2_{xx}H(x) + \partial_x H(x) \;=\; \partial_x b(x) \text{ in } (0, L) \qquad (9.20)$$

with $\Lambda_{ref} = \frac{(H_{ref}-b_{ref})}{|\partial_x H_{ref}|}$. (H_{ref}, b_{ref}) are given functions such that $h_{ref}(x) = (H_{ref} - b_{ref})(x) \geq h_{min} > 0$ a.e.

The equation is closed with non-homogeneous Dirichlet boundary conditions.

The inverse problem as those presented in Section **??** is considered: infer $b(x)$ given somme measurements $H^{obs}(x)$.

The unknown parameter $b(x)$ appears in the RHS of the equation only (through its derivative): this case fits the general basic case addressed in Section 9.3.4.

Let us solve this inverse problem by a variational approach.

We set the objective function J_α as follows:

$$J_\alpha(b; H) \; = \|H - H^{obs}\|_2^2 + \alpha_{reg}\|b - b_b\|_2^2,$$

with b_b a background value (first guess).

The inverse problem is solved as computing:

$$b^*(x) = arg \min_{b(x)\in\mathcal{B}} j_\alpha(b(x)),$$

with $j_\alpha(b) = J_\alpha(b\,; H^b)$, H^b the (unique) solution of (9.20) given b.

Exercise 9.1.
Write in a synthetic way the optimality system which characterizes the solution b^* of this optimal control problem.

The local gradient values: a potential interesting information Computing a gradient, even without performing a minimization algorithm, may present some interests in a modeling point of view. Indeed, the gradient value represents a local sensitivity of the model output j with respect the parameters u. These gradient values may help to understand the parameter influences on the output criteria j, especially if it represents spatially distributed information. Nevertheless, such sensitivity analyses remain limited since local in the sense that it is regarding at a given point u only (and regarding to the considered observations too).

Let us remark that this idea of computing sensitivities in order to better understand both the model and the physics *can be applied in a context without observations.*
Typically, it can be applied to cost functions depending on the state of the system only in view to study the system stability.
For example, it can be interesting to quantify the sensitivity of the following model output:

$$j(u) = \frac{1}{2}\int_\omega \|\nabla y^u\|^2 dx$$

with ω a subset of Ω.

Chapter 10

Optimal Control of ODEs

Optimal control is the foundation of automation, which is widely developed in fields such as aeronautics, robotics, etc. Optimal control fundamentally relies on (differentiable) optimization. In this chapter, the optimal control of systems governed by an Ordinary Differential Equation is briefly studied. The mathematical material presented in this chapter closely follows the presentation done in [39]. (The reader may refer to the excellent course manuscript [1], however in French).

We focus on the case of a Linear ODE and a Quadratic cost function, that is the so-called LQ system, since the complete analysis can be written and enables to illustrate in detail the approach. A proof of existence and uniqueness of the optimal solution is provided; the necessary (here sufficient) conditions of optimality are detailed. The latter are characterised as the critical points of the *Hamiltonian*. The *Pontryagin's minimum principle* which states the result is introduced.

The two classes of numerical approach to solve optimal control problems are presented, namely the direct approach and the indirect one. The latter relies on the two-point value problem deriving from the Pontryagin's principle.
A central goal of the textbook is Variational Data Assimilation. As a consequence, we mainly focus on open-loop controls and not so much on closed-loop controls (based on the Riccati equations) despite closed-loop controls are the key concept for systems' control. For the same reason, the notion of controllability (and observability) are very briefly presented only too. We refer for example to [19, 39, 7] for a full course on the subject.

Optimal control also at the basis of more contemporary methods such as *Reinforcement Learning (RL)*, one of the most promising method in Artificial Intelligence. While optimal control relies on a well-determined, physics-based model, RL can be used in situations where the system model is not fully known. Moreover, RL relies on stochastic processes, more precisely on Markov Decision Processes, which are discrete-time optimal stochastic control problems.

The outline of this chapter is as follows[1]

Contents

[1] Recall that the sections indicated with a * are ”to go further sections”. They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

10.1 Example: dynamic control of a vehicle

We here consider the basic example of a vehicle dynamic following a 1D trajectory. (The vehicle could be e.g. a train or a trolley). The reader can find other examples e.g. in [41, **?**].

This problem is numerically solved by the Python code provided on the INSA Moodle page of the course.

10.1.1 The model

Model equation in the original position variable $x(t)$

The problem here considered is as follows. At instant t, the vehicle position is represented by $x(t)(m)$, its velocity by $x'(t)(ms^{-1})$.
The goal is to control the vehicle trajectory by acting on a command $u(t)$ e.g. the engine power. We have $x : [0,T] \to \mathbb{R}$ and $u : [0,T] \to \mathbb{R}$.
Let m (kg) be the vehicle mass, u be the pedal position (in %).
The dynamic trajectory model may simply reads as:

$$m\,x"(t) = -K\ x'(t) + G\ u(t) \text{ in } (0,T) \tag{10.1}$$

with K a friction coefficient (Ns/m) and G a gain parameter $(ms^{-1}(\%\text{pedal})^{-1})$.
The equation is closed with I.C.: $(x', x)(0)$ is given e.g. equal to $(0,0)$.

Model equation in variable $y(t) = x'(t)$ (velocity variable)

By making the change of variable $y(t) = x'(t)$, the state equation simply reads:

$$y'(t) = -k_1 y(t) - k_2\ u(t) \text{ in } (0,T) \tag{10.2}$$

with $k_\square$ constant parameters of the model. The direct model is then a simple 1st order linear ODE in the velocity variable y. The equation is closed with the I.C. $y(0) = 0$. Given $u(t)$, the equation has an unique solution $y(t)$.

The original variable of position x can be next recovered as: $x(t) = \int_0^t y(s)ds + c$ with c s.t. $x(0) = 0$.

10.1.2 Inverse problems

A few interesting questions are the following ones. All of them can be formulated as inverse problems, which are here optimal control problems.

- Given a target velocity z_{target}, identify a value of $u(t)$ such that the vehicle reaches this velocity value z_{target} at time T given, by consuming a minimum of energy (minimal value of u).

- A more dynamic version of the problem would be as follows. Given a target velocity $z_{target}(t)$, identify a value of $u(t)$ such that the vehicle sticks as close as possible to $z_{target}(t)$, moreover by consuming a minimum of power, moreover by considering bounded accelerations, etc.

One can mathematically translate the inverse problem above by minimizing the following cost function:

$$j_\alpha(u) = \frac{1}{2}\int_0^T |y^u(t) - z_{target}(t)|^2 dt + \alpha\frac{1}{2}\int_0^T \|u(t)\|_N^2 dt \tag{10.3}$$

with y^u (equivalently $y(u(t))$) the unique solution of the model equation, given u.

N denotes a symmetric positive matrix therefore defining a semi-norm. If moreover definite, N defines a norm.

The optimal control problem reads:

$$\min_{u(t)\in\mathcal{U}} j_\alpha(u) \tag{10.4}$$

where $\mathcal{U}$ is a (semi-)interval of $\mathbb{R}$, representing potential inequality constraints on u.
In higher-dimensions, $m > 1$, $\mathcal{U}$ has to be a convex closed subset of $\mathbb{R}^m$.
The optimization problem (10.4) is bi-objective: a trade-off between the two terms have to found. The two cost function terms are balanced through the weight parameter α, $\alpha > 0$.

In the case the objective is to do not accelerate excessively, then we may define the term in u in (10.3) as: $\frac{1}{2}\int_0^T |u'(t)|^2 dt$. In this case N is defined as: $\|v\|_N = |\frac{d^l}{dt^l}v|_2$ with $l = 0, 1$ or even 2.

This optimal control problem will be solved in Section 10.4.3 by employing a simple numerical direct method.

Remarks

- The cost function $j(u)$ depends on u explicitly through its second term, but also through $y^u(t)$ in its first term: this is why it is a optimal control problem and not simply a standard optimization problem.

- If considering the regularization term $|u|^2$ in $j(u)$ and since the model is linear, then the cost function $j(u)$ is strongly convex.
  In this case, the problem is what we call a *Linear-Quadratic (LQ) optimal control problem.* It will be shown later that LQ problems admits an unique solution u^* (under assumptions on $\mathcal{U}$).

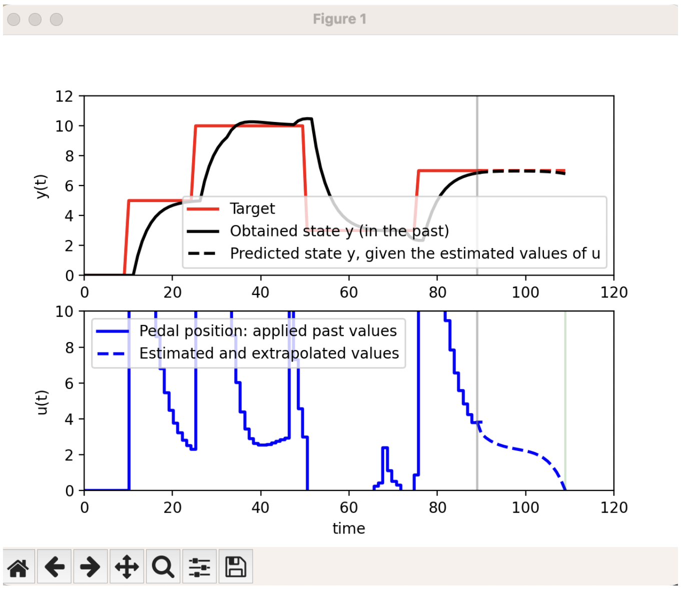


Figure 10.1: Optimal control of a vehicle: given a velocity target $z_{target}(t)$, what is the optimal control value $u(t)$ while imposing "reasonable" accelerations ? Figure plotted by the ocmputational code provided on the webpage of the course.

10.2 Introductory remarks

10.2.1 Control theory in a nutshell

The control theory aims at analyzing dynamical systems modified by a command (the control). Control theory refers to two classes of objectives.

A) *Controllability problems* aims at acting on the control in order to bring the system state to a final state given (if possible). The goal is generally to stabilize the system by feedback in order to make it insensitive to perturbation. Controllability is a central (hot) topic in automation. It is however not a shared goal with Data Assimilation (DA). As a consequence, controllability of systems is not discussed in this textbook.

B) *Optimal control problems* aim at defining a control u such that a criteria (the cost function) $j(\cdot)$ is minimal. The cost function $j(\cdot)$ depends on the control but also on the state of the system.
A general formulation of an optimal control problem may be written as follows.
We consider the dynamical system

$$\dot{y}(t) = F(t, y(t), u(t)) \text{ in } (0, T) \tag{10.5}$$

where $y : t \mapsto y(t) \in \mathbb{R}^d$ represents the state of the system, $u : t \mapsto u(t) \in \mathbb{R}^m$ represents the control, and $F : (t, y(t), u(t)) \in (0, T) \times \mathbb{R}^n \times \mathbb{R}^m \mapsto F(t, y(t), u(t) \in \mathbb{R}^m$ represents the dynamic of the system.

We consider the functional j, $j : u \in \mathbb{R}^m \mapsto j(u) \in \mathbb{R}$, defined as:

$$j(u) = \int_0^T \|y(u(t)) - z(t)\|_W^2 dt + \|y(T) - z_T\|_Q^2 + \int_0^T \|u(t)\|_U^2 dt \tag{10.6}$$

with $y(u(t))$ solution of (10.5) given $u(t)$. The function $z(t)$ and z_T are given targets. W, Q and U denote semi-norms and a norm, respectively.

The optimal control problem consist to find $u^*(t) \in \mathcal{U}$ satisfying

$$u^* = arg \min_{u(t) \in \mathcal{U}} j(u) \tag{10.7}$$

where $\mathcal{U}$ is a convex closed subset of $\mathbb{R}^m$, potentially defining inequality constraints on u.

Optimal control is a science between the automation science and applied mathematics. Optimal control formulation is at the basis of Variational DA formulations too. Bases of optimal control are developed in chapters 10 and 11. For extended presentations to optimal control theories, the reader may consult e.g. the excellent books [19, 39].

10.2.2 ODE solution behaviours: simple examples

The aim of this section is simply to illustrate how the response of an apparently gentle scalar second order ODE equation can widely change if simply changing the source term expression. The source term can be seen as the command of the system.

The simple ODE-based model: spring - mass system

Let us consider here a similar basic example: the dynamics of a spring - mass system, see Fig. 10.2.
The mass m is submitted to a force $f(y)$, which is supposed here to be equal to:$-[k_1(y-L) + k_2(y-L)^3]$.
L is the spring length at rest, $k_\Box$ are the spring stiffness parameters.
An external force (depending on time t) is applied: $u(t)\vec{i}$.

Given the external force $u(t)$, the spring position $y(t)$ is described by the differential equation:

$$my''(t) + k_1(y(t) - L) + k_2(y(t) - L)^3 = u(t) \text{ for } t \geq 0$$

This 2nd order ODE is closed with the I.C. $(y, y')(0) = (y_0, y'_0)$ given.
It is a linear ODE if $k_2 = 0$, non-linear if not.
$u(t)$ can be perceived as a *control* of the system.

For sake of simplicity, without changing the nature of the problem, we set: $m = 1, k_1 = 1, L = 0$.

Different states of the system according to different control values

Case $u(t) = 0$ i.e. without any external action.
In this case the equations reads:

$$y''(t) + y(t) + 2y(t)^3 = 0$$

It is a particular case of the Duffing equation.
Its solutions $y(t)$ satisfy: $y(t)^2 + y(t)^4 + y'(t)^2 = c$, c a constant.
All solutions are periodic and can be represented by an algebraic curve. The phase diagram and the trajectory are plotted on Fig. 10.2.

Case $u(t) = -y'(t)$**.** In this case, the applied external force aims at damping the spring. The equation reads:

$$y''(t) + y(t) + 2y(t)^3 + y'(t) = 0$$

The numerical solution is computed, then the phase diagram and trajectory are obtained, see Fig. 10.2.

Using the Lyapunov theory, it can be shown that the origin is asymptotically stable. The spring position and the velocity reach the equilibrium position in infinite time, not in finite time.

Case $u(t) = -(y(t)^2 - 1)y'(t)$**.** With this control expression, the model is a particular case of the classical *Van der Pol* equation:

$$y''(t) + y(t) + 2y(t)^3 + (y(t)^2 - 1)y'(t) = 0$$

Two different solutions are computed and plotted on Fig. 10.2 (phase diagram and trajectories).
Using the Lyapunov theory, it can be proved that it exists a periodic solution which is attractive (plotted on Figure 10.2).

This very classical example is treated in detail in [39], we refer to this book for more details.

These three examples simply illustrate the wide range of behaviours which can be obtained by simply changing the control expression, even in the case of a gentle scalar equation.

10.2.3 On the controllability of a system*

This section is a "to go further section".

When addressing an optimal control problem, a natural question is: whether or not the control can make the system reach the target ?

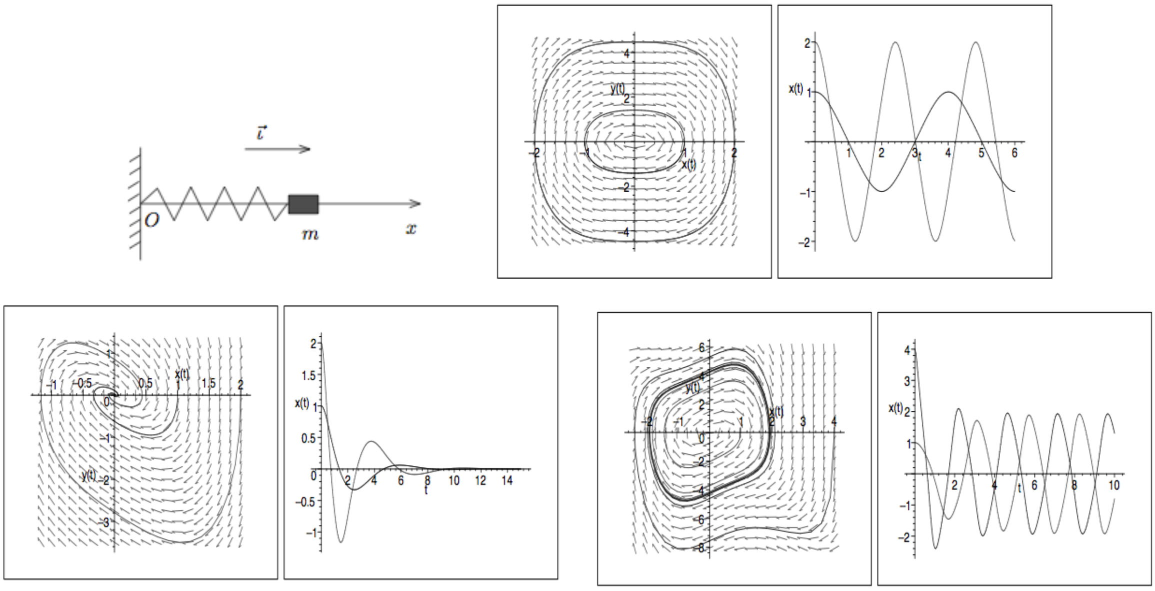


Figure 10.2: The spring-mass system example from [39]. (Up)(L) The spring-mass system. (Up)(R) The solution (state of the system) without control. (Down) The solution: (L) with damp control, (R) with another control value leading to the Van der Pol equation.

This question of controllability is important in a context of automation of course. It is not so much in a context of Data Assimilation. However, we briefly present below the concept of controllability for a linear dynamical system. The reader may refer e.g. to [39, **?**] for details.

Concept of reachable set

Let $y_u(t)$ be the solution of the ODE system corresponding to a given control $u(t)$ (y_u is supposed to exist and to be unique). The set of reachable states from the initial state y_0 in time $T > 0$, can be defined as follows:

$$Acc(y_0, T) = \{y^u(T), \text{ with } u \in L^\infty([0, T], \mathbb{R}^m)\}$$

with $Acc(y_0, 0) = y_0$.
Here, the control variable is not constrained: $u \in \mathbb{R}^m$.

We say that the system is controllable at any time T if $Acc(y_0, T) = \mathbb{R}^n$. That is for any y_0 and z_T in $\mathbb{R}^n$, there exists a control $u(t)$ such that $y^u(0) = y_0$ and $y^u(T) = z_T$.

Controllability of autonomous linear systems: the Kalman condition

The system is said to be autonomous if A and B do not dependent on time, that is we have the dynamical system:

$$y'(t) = Ay(t) + Bu(t) + r(t)$$

The Kalman condition is a remarkable algebraic result which states that the autonomous linear ODE above is controllable at any time T if and only if the rank of matrix $C = (B, AB, ..., A^{n-1}B)$ equals n.

This matrix C is called the Kalman matrix. The condition $\text{rank}(C) = n$ is called *the Kalman controllability condition.*

Note that the Kalman condition does not depend neither on T nor on y_0. **ToDo: COMMENT, cf poly Allaire**

Of course, this results does not hold for general systems, in particular non linear ones.

10.3 The Linear-Quadratic (LQ) problem

This section addresses the reference optimal control problem class: the Linear-Quadratic (LQ) problem. The model is linear (in the state) if the operator $y(t) \mapsto F(t, y(t), u(t))$ in (10.5) is linear.

10.3.1 The general linear ODE model

Let $A(t)$, $B(t)$ and $S(t)$ be three linear operator defined from $I =]0, T[$ into $\mathcal{M}_{n,n}(\mathbb{R})$, $\mathcal{M}_{n,m}(\mathbb{R})$ and $\mathbb{R}^n$ respectively. These three mappings are assumed to be bounded: they belong to $(L^{\infty}(I))^{\square}$. (This assumption could be relaxed since locally integrable would be sufficient).
We consider the following linear first order ODE:

$$\boxed{\left\{\begin{array}{l}\text{Given } u(t), \text{ find } y(t) \text{ such that:}\\ y'(t) = A(t)y(t) + B(t)u(t) + S(t) \text{ for } t \in I = [0, T]\\ \text{with the initial condition } y(0) = y_0.\end{array}\right.} \tag{10.8}$$

The control $u(t)$ is assumed to be bounded: $u(t) \in (L^{\infty}(I))^m$.
Note that the command operator B is here linear in u: the map $u(t) \mapsto B(t)u(t)$ linear.

Explicit expression of the solution $y^u(t)$

A classical result states that (10.8) has one and only one solution $y(t)$, $y(t)$ continuous from I into $\mathbb{R}^n$. Moreover an explicit expression of y in integral form holds.
Let us consider for sake of simplicity the case $S = 0$. In this case, we have:

$$y(t) = M(t)y_0 + M(t) \int_0^t M(s)^{-1} B(s)u(s)ds \tag{10.9}$$

with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t), \;\; M(0) = Id$.
Note that if $A(t) = A$ constant then $M(t) = \exp(tA)$.

The control-to-state map $\mathcal{M}(u)$

The general *optimal control problem* will consist to find a control $u(t)$ *minimizing* a given criteria (cost function) $j(u)$.

Let us introduce the control-to-state map (model operator) $\mathcal{M}(u)$:

$$\mathcal{M} : \;\; u(t) \mapsto y^u(t) \equiv y(u(t))$$

with $\mathcal{U} \subset L^\infty(I, \mathbb{R}^m)$ and $\mathcal{Y} \subset C^0(I, \mathbb{R}^n)$.

The following central property holds.

Proposition 10.1. *In the linear case, the control-to-state operator $u(t) \mapsto \mathcal{M}(u(t))$ is affine, for all t in $[0, T]$.*

Proof. Let $y(t)$ be the (unique) solution associated to $u(t)$: $y(t) = \mathcal{M}(u(t))$. The result follows straightforwardly from the explicit expression (10.9) of the solution $y(t)$, here in the case $S = 0$. In the general case with $S \neq 0$, the expression is more complex but the argument remains the same. □

10.3.2 Quadratic objective function

The choice of the cost function to be minimized is part of the problem definition. Since it will be minimized, convexity properties are expected.
Moreover if using computational gradient-based methods, differentiability properties are expected too.

A typical objective function reads as:

$$\boxed{J(u; y) = \frac{1}{2} \int_0^T \|y(t) - z_d(t)\|_W^2 dt + \frac{1}{2} \int_0^T \|u(t)\|_U^2 dt + \frac{1}{2} \|y(T)\|_Q^2} \tag{10.10}$$

with $z_d(t)$ a given target.
Note that for sake of simplicity and without loss of generality, the target value $z_d(t)$ can be set to 0. Moreover each term weight has here been set to 1 (compared to 10.6).

The cost function $j(u)$ is next defined from the objective function $J(u; y)$ by:

$$\boxed{j(u) = J(u\,; y^u)} \tag{10.11}$$

with y^u the unique solution of the (linear) model, given u.

The three terms are the time averaged cost of the state, the control and the final state, in the semi-norms W, U and Q respectively.
This functional j is a multi-objective cost functional.

By construction, the maps $u \mapsto J(u; \cdot)$ and $y \mapsto J(\cdot; y)$ are quadratic with respect to u and y respectively.
Therefore the objective function $J(u; y)$ is obviously differentiable, moreover strictly convex.
Indeed, "Quadratic $\Rightarrow C^\infty$ and strictly convex".

On the contrary, the cost function $j : u \mapsto j(u)$ is *not* quadratic.
However, since the operator $\mathcal{M} : u \mapsto y^u$ is affine (Prop. 10.1), $j(u)$ is strictly convex.
Indeed, roughly speaking: quadratic $\circ$ affine $\Rightarrow$ convex, and strictly convex under assumption on the definiteness of the Hessian. This rough assertion has to be investigated more in details.

Mathematical considerations. The operators Q, W and U are symmetric positive matrices in $\mathcal{M}_{n,n}(\mathbb{R})$, $\mathcal{M}_{n,n}(\mathbb{R})$ and $\mathcal{M}_{m,m}(\mathbb{R})$ respectively, therefore defining semi-norms. Moreover U is supposed to be definite. Thus, the penality term in u is minimal for u vanishing.

Let us recall that for a matrix M symmetric positive definite, in vertu of Courant-Fischer's theorem (Rayleigh's quotient), there exists a constant $c > 0$ such that: $\forall v \in \mathbb{R}^m$, $\|v\|_M^2 \geq c\|v\|^2$. In other words, such linear operator M is coercive, uniformly in time.

Since the observation functional is quadratic and the model is linear, the natural functional space for control variable is $M = L^2([0,T], \mathbb{R}^m)$.
Let us point out that the a-priori natural space $C^0([0,T], \mathbb{R}^m)$ is not a Hilbert space...

In practice, W, U and Q are often diagonal positive matrices whose the diagonal coefficients are tuned to define the relative importance of the terms.

10.3.3 The Linear-Quadratic (LQ) optimal control problem

The optimal control problem defined by (10.8)-10.10) reads as follows.
Given y_0 and T, find $u^*(t)$ such that

$$\boxed{u^* = arg \min_u j(u)} \tag{10.12}$$

with $j(u) = J(u\,; y^u)$, $y^u(t)$ the solution of the linear ODE (10.8).

It is a *Linear-Quadratic (LQ) optimal control problem.*

The LQ problem is quite idealistic. However, many instructive properties of the system can be written, both in a mathematical and numerical point of view. To better understand more complex non-linear problems or non quadratic observation function $J(u; y)$, a complete analysis of the LQ problem provides good insights.

10.3.4 Existence and uniqueness of the solution in the LQ case *

This section is a "to go further section".
It is shown below that under the presence of the term $\int_0^T \|u(t)\|_U^2 dt$ in the objective function, it exists an unique solution u^* to the problem (the cost function in this case is strictly convex).

For a sake of simplicity, we do not consider here the source term in the direct model: $S(t) = 0$.

Theorem 10.2. *Let $j(u)$ be defined by (10.10). It exists a unique $u \in M$ minimizing $j(u)$ with the "constraint" $y(t)$ solution of (10.8).*
In other words, it exists a unique optimal control $u(t)$ and a corresponding trajectory $y(t)$ to the LQ problem.

Proof derived from those presented e.g. in [39].
A) Existence.
It is based on the convergence of minimizing sequence (calcul of variations, D. Hilbert, 1900 approx.).
Step 1). The cost function is bounded by below: $\inf\{j(u), u \in M\} > -\infty$ since $j(u) \geq 0$. There exists a minimizing sequence (u_n) defined for all $t \in [0, T]$; i.e. a sequence such that:

$$\lim_{n \to +\infty} j(u_n) = \inf\{j(u), u \in M\}$$

(As a matter of fact, $\forall n \in \mathbb{N}$, $\exists v_n$ such that: $m \leq j(v_n) < m + \frac{1}{n}$).

Step 2) There exists $\alpha > 0$ such that: $j(u) \geq \alpha \|u\|_M^2$. Thus, the minimizing sequence (u_n) is bounded in M. Hence there exists a sub-sequence (u_{n_k}) which converges weakly to a control u in M:

$$u_{n_k} \rightharpoonup u \text{ in } L^2(I)$$

Step 3) Let us denote by y_n (resp. y) the state associated to u_n (resp. u). The system (10.8) is a first order linear O.D. E. (and with $S(t) = 0$); we known an explicit expression of the solution:

$$\forall t,\ y_n(t) = M(t)y_0 + M(t) \int_0^t M(s)^{-1} B(s) u_n(s) ds \qquad (10.13)$$

with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t)$, $M(0) = Id$. (If $A(t) = A$ constant then $M(t) = \exp(tA)$).

Similarly, we have: $\forall t,\ y(t) = M(t)y_0 + M(t)\int_0^t M(s)^{-1}B(s)u(s)ds$. Thus, we obtain that the sequence $(y_n)_n$ converge to y.
Passing to the limit in (10.13), we obtain the existence of y_u, solution corresponding to u.

Step 4) It remains to prove that u minimizes j. Since $u_n \rightharpoonup u$ in L^2, since j is continuous hence lower semi-continuous, we have (by definition):

$$j(u) \leq \liminf_n j(u_n)$$

and necessarily $j(u) = \inf\limits_{v\in M} j(v)$.

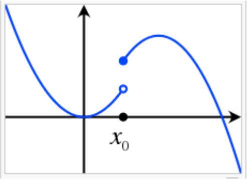


Figure 10.3: A function j lower semi-continuous at u_0: for u close to u_0, $j(u)$ is either close to u_0 or lower than u_0.

In other words, $u(t)$ minimizes $j(u)$ ($j(u) = \min\limits_{v\in M} j(v)$) and the corresponding state (trajectory) y_u is an optimal trajectory.

B) Uniqueness.
Recall that we here consider the cost function defined as follows, see (10.10)(10.12):

$$j(u) = \frac{1}{2}\int_0^T \|y^u(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + \frac{1}{2}\|y^u(T)\|_Q^2$$

For all t, $\|u(t)\|_U$ is a norm hence convex but not strictly convex. Proof of this assertion: the triangle inequality...
However, in a Hilbert space, the square of a norm is strictly convex, see e.g. [5] Chap. 10, p118. *Proof: write the second derivative of the function $q : u \mapsto \|u(t)\|2_U =< UU^Tu, u >$).*
As a consequence the term $\|u(t)\|_U^2$ implies strict convexity of the $j(u)$:
$\forall (u_1, u_2) \in M^2,\ \forall t \in]0, 1[$,

$$j(tu_1 + (1-t)u_2) < tj(u_1) + (1-t)j(u_2)$$

unless $u_1 = u_2$.
Moreover, it has been previously proved that the control-to-state operator $\mathcal{M}(u)$ is affine therefore convex, see Prop. 10.1.
In other respects $\|.\|_W$ and $\|.\|_Q$ are semi-norms therefore convex.
Finally, the cost function $j(u)$ is strictly convex and the uniqueness follows.
Indeed, let u_1 and u_2 be such that: $j(u_k) = \inf\limits_{v\in M} j(v),\ k = 1, 2$.

We have: $j(tu_1 + (1-t)u_2) < tj(u_1) + (1-t)j(u_2)$.
Hence: $j(tu_1 + (1-t)u_2) < \inf_{v \in M} j(v)$ unless $u_1 = u_2$, which must be the case. □

Remark 10.3.
- The strict convexity of $j(u)$ is due to the quadratic term $\|u(t)\|_U^2$. If not present, $j(u)$ is convex only and the uniqueness does not hold anymore.

- This term $\|u(t)\|_U^2$ can be perceived as a Tykhonov regularization term.

- In the autonomous case (A and B constant), we have:

$$\|y'(t)\| \leq \|A\| \|y(t)\| + \|B\| \|u(t)\| \leq cst(\|y(t)\|^2 + \|u(t)\|^2)$$

Then, if all hypothesis are satisfied in $I = [0, +\infty$ then $y'(t)$ is in $L^1(I)$ and necessarily the minimizing trajectory $y(t)$ tends to 0 when t tends to $+\infty$.

10.4 Numerical methods for optimal control problems

10.4.1 Two classes of numerical methods: direct, indirect

Basically, it exists two classes of numerical methods to solve an optimal control problem (whatever if linear or not): the *direct methods* and the *indirect methods.*

- *Direct methods* simply consist in discretizing the state y and the control u, to reduce the problem to a standard discrete optimization problem.
  Next, the minimization relies on nonlinear programming algorithms such as e.g. the classical Sequential Quadratic Programming (SQP) algorithm.
- *Indirect methods* consist in numerically solving a problem resulting from the so-called *maximum principle.*
  The latter relies on the *necessary first-order optimality* conditions and on the *Hamiltonian.* These concepts are presented in next sections.

Methods mixing the two approaches are frequently employed too.

In short, pros and cons of each approach are as follows.

- o Direct methods are easy to implement. They do not require to introduce the Hamitonian and the adjoint equation (which are introduced in the next sections). They are tractable in small dimensions only, not in large dimensions.

- o Indirect methods require to derive the optimality system relying on the Hamiltonian and the adjoint equations (see next sections). They are efficient in all dimensions, large ones included.

Indirect methods are presented after having introduced the Pontryagin principle and derived the optimality system which is based on the Hamiltonian concept.

10.4.2 Direct methods

Direct methods consist to:

- write the optimal control problem equations in a discrete form,
- solve the optimization problem by a standard differentiable optimization algorithm e.g. the classical Sequential Linear Quadratic (SQL) algorithm.

Let us write in a discrete form the general LQ problem (10.8)(10.10). A regular time grid $(t_0, \cdots, t_N)$ is considered (constant time step h). The discrete state y_h, $y_h = (y^1, \dots, y^N)$, is obtained by performing a numerical scheme e.g. Runge-Kutta 4 (or Euler scheme).
In discrete form, the problem to be solved reads:

$$\text{argmin }_{(u_1,\cdots,u_M)} j(u_1, \cdots, u_M) \tag{10.14}$$

with $u_h = (u_1, \dots, u_M)$, $j(u_h) = J(u_h; y_h(u_h))$.

For a sake of simplicity, let us consider the basic explicit Euler scheme to solve the equation $y'(t) = A(t)y(t) + B(t)u(t)$.
The discrete system reads:

$$y^{n+1} = (1 + hA(t^n))y^n + hB(t^n)u^n \text{ for } n = 0 \cdots (N-1); \quad y^0 = y_0 \tag{10.15}$$

The finite dimensional optimization problem reads:

$$\boxed{\left\{ \begin{array}{c} \min_{u_h=(u_1,\cdots,u_M)} j(u_h) \\ \text{under the } N \text{ constraints (10.15) on } y_h = (y^1, \dots, y^N) \\ (= \text{the numerical scheme equations}) \\ \oplus \text{ potential equality-inequality constraints on } u_h. \end{array} \right.} \tag{10.16}$$

Such (finite dimensional) optimization problem, with (equality-inequality) constraints, can be solved by the Sequential Quadratic Programming (SQP) algorithm.
Other variants of formulations can be considered.

Recalls on the SQP algorithm Sequential Quadratic Programming (SQP) algorithms denote iterative methods for constrained nonlinear optimization problems. The objective function and the constraints are supposed to be C^2.

The SQP algorithms principle is as follows. One solves a sequence of optimization subproblems; each of them optimizes a *quadratic representation of the objective function*, under the constraints which are linearized.

- If the problem is unconstrained, then the method reduces to the Newton method: the optimum is such that it makes vanish the gradient.

- If the problem has equality constraints and no inequality constraints, then the method is equivalent to apply Newton's method to the first-order optimality conditions (KKT condition).

Pros and cons are as follows.

⊕ The algorithm is available on any well built optimization library or computational system.

⊖
- Each iteration may demand a lot of computational time.
- The algorithm requires cost functions twice differentiable.
- The algorithm may converge to local minima only.

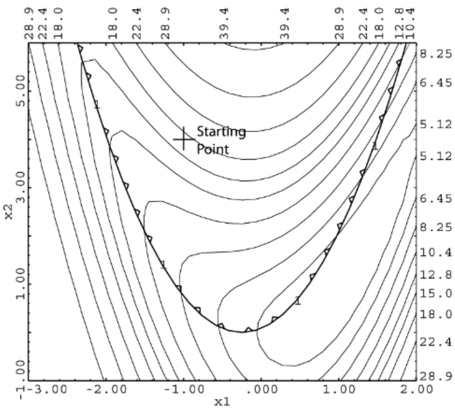


Figure 10.4: The SQP algorithm for an example. Image extracted from [].

10.4.3 Numerical solution of the optimal vehicle dynamic

A Python code numerically solving the 1D example presented in Section 10.1 is available on the course webpage.

Since the problem is of tiny dimension (it is 1D only), and the model is very low CPU-time consuming (a 1D scalar ODE), a basic direct method is well adapted.

Exercise 10.4. *1) Detail the equations to solve by a direct method the vehicle dynamic problem described in Section 10.1.*
2) Detail the numerical algorithm implemented into the Python code provided on the Moodle course page. Follow the instructions (supplementary material).

A correction is presented in the next two paragraphs.

1) The discrete equations

A good choice of time scheme is the Runge-Kutta RK4 scheme. However, for a sake of simplicity, let us consider here the explicit (forward) Euler time scheme.
The time interval $[0, T]$ is digitalized using a constant time step: $h = T/N$, $t_i = ih$, $i = 0, \ldots, N$.
Given the I.C. y_0, we have for $n = 0 \cdots (N-1)$,

$$y^{n+1} = y^n - h\left(k_1 y^n + k_2 u^n\right) \tag{10.17}$$

Given the target velocity z_{target} and considering minimal variations of the command, we set:

$$j_\alpha(u) = \frac{1}{N}\sum_n |y^n - z_{target}(t^n)|^2 + \alpha\frac{1}{N}\sum_n (u'(t^n))^2$$

And we solve the optimization problem:

$$\text{argmin }_{u=(u_1,\cdots,u_N)} j(u)$$

with the N equality constraints (10.17).

Additional inequality constraints on u, or even equality constraints on some values of u_n, also inequality constraints on the state y.

2) The coded algorithm

The time grid described below is those of the command. The time grid to solve the model equation is (much) smaller; it is not explicitly detailed. The current time step index is denoted by i.
The control is performed on sub-intervals centred on i therefore considering fast and future time intervals. The time interval length (the horizon) is denoted by M: $I_M = [i - M, i + M]$.
The model equation is then solved on a sub-interval centred on i of same length M or potentially larger one of length P: $I_P = [i - P, i + P]$ with $P \geq M$. The ODE solver is the standard RK4 scheme.

A possible algorithm version is as follows.
This is the algorithm which has been coded into the Python program available on the course page.

- Initialization. The state $y(0)$ is given (I.C.).
- For each time instant i (of the command time grid), $i = 1, \ldots, N$,
  - Setup the current working sub-interval I_P, $I_P = [max(0, i - P), i + P]$ (P the prediction horizon),
  - Given the values of control at time instant t_{i-1} and t_i, solve the ODE in the time step interval $[t_{i-1}, t_i]$. This provides the state at current time, $y_i \approx y(t_i)$.
  - Evaluate the cost function in the prediction interval i.e. $j_\alpha(u_{i+1}, \ldots, u_{i+P})$,
  - Solve the (standard) optimization problem:

$$\min_{(u_i, \ldots, u_{i+M})} j_\alpha(u_i, \ldots, u_{i+P}) \tag{10.18}$$

  Note. For the indices in $\{i + M + 1, \ldots, i + P\}$, the control values are here set to the value at index $(i + M)$ i.e. $u^{(i)}(i + M)$.

  This step provides the control value $u^{(i)}$ in the current working sub-intervall $[t_i, t_{i+P}]$.

- Iterate

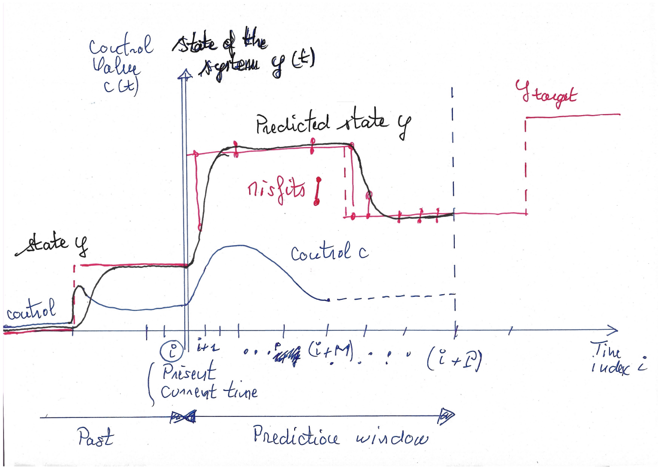


Figure 10.5: The optimal control policy is here simply imposed by using a basic direct method since the problem is of small dimension (a scalar control variable) and the model is very low CPU-time consuming.

10.5 The Pontryagin principle & Hamiltonian (for open-loop control)

In this section are presented the fundamental concepts of Pontryagin's principle, Hamiltonian, optimality system which include the adjoint equation.

The Pontryagin's principle states a necessary optimality condition. This necessary condition is sufficient in the LQ case.

Moreover, as previously shown, direct numerical methods do not require the use of any new concept such as the Hamiltonian. However, the resulting algorithms are tractable in small dimensions only.

To address large dimensions systems, typically $y(t) \in \mathbb{R}^n$ with $n = O(10^p)$, $p \approx 4$ and more, indirect methods may be required. The latter relies on the optimality system which need the introduction of the Hamiltonian and the adjoint equation concepts.

Open-loop control vas closed-loop control The Pontryagin's principle states optimality condition useful for open-loop control, Fig. 10.6.
Recall that in automation, the expected information is a closed-loop control law tant is including a feedback law. Such feedback law is not required in a Data Assimilation context. In the LQ case, the feedback laws is known and relies on the Riccati equation, see Section 10.6. This section addressing the feedback law required for closed-loop control, can be skipped for readers interested in DA only.

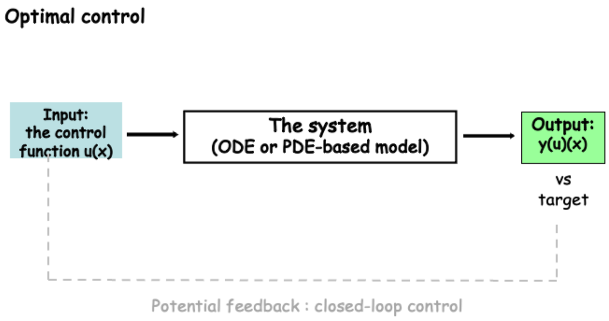

Optimal control
Input: the control function u(x)
The system (ODE or PDE-based model)
Output: y(u)(x)
vs target
Potential feedback : closed-loop control

Figure 10.6: Optimal control of a system: open loop control vs closed loop control.

10.5.1 The Pontryagin minimum principle

L. Pontryagin, Russian mathematician, 1908-1988.

In the case of *non-linear* state equation, the cost function is non-convex and the Pontryagin minimum principle (also called maximum principle) states a *necessary condition* of optimality.

In the LQ case, the Pontryagin minimum principle is a *necessary and sufficient condition* of optimality. The Pontryagin minimum principle relies on the concepts of Hamiltonian[2] and *adjoint equation.*

[2] The so-called "Hamiltonian" in control theory is a particular case of the Hamiltonian in mechanics; it is inspired by the Lagrangian you have studied during your optimization course.

Let us recall the equations in the LQ case, see (10.8)(10.10). The model reads:

$$\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) + S(t) \text{ for } t \in I = [0,T] \end{cases} \tag{10.19}$$

with I.C.: $y(0) = y_0$. Next the cost function j satisfies $j(u) = J(u; y^u)$ where y^u is the unique solution of the model and $J(\cdot;\cdot)$ is the quadratic observation function defined by:

$$J(u;y) = \frac{1}{2}\int_0^T \|y(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + g(y(T)) \tag{10.20}$$

Note that we consider here the term in $y(T)$ slightly more general than before with $g(\cdot)$ any function defined from $\mathbb{R}^n$ into $\mathbb{R}$, C^1 and convex.

The goal is to characterize the optimal control satisfying: $u^* = arg\min_u j(u)$ with $j(u) = J(u; y^u)$.

Theorem 10.5. *The trajectory $y(t)$ associated with the control $u(t)$, is optimal for the LQ problem (10.19)(10.20) if there exists an adjoint field $p(t)$ which satisfies:*

$$-p'(t) = p(t)A(t) - y^T(t)W \text{ for almost } t \in [0,T] \tag{10.21}$$

with the Final Condition: $p(T) = -\nabla g^T(y(T))$.
By convention, $p(t)$ is here a line vector, on the contrary to $y(t)$.
Furthermore, the optimal control u satisfies:

$$Uu(t) = B^T(t)\ p^T(t) \text{ for almost } t \in [0,T] \tag{10.22}$$

Remark 10.6.

- *Note that Eq. (10.21) written in $q \equiv p^T$, q a column vector as y, reads:*

  $$-q'(t) = A^T(t)q(t) - Wy(t)$$

  which better highlights the terminology "adjoint equation".

- *This theorem provides an expression of the optimal control u^* not explicitly depending on the state y but in function of an auxiliary field p called the adjoint field.*
  The adjoint field $p(t)$ is solution of a linear equation "similar" to the model one with $-A^T$ instead of A in particular. it is therefore reverse in time, therefore starting from a condition at $t = T$. p depends on y through the adjoint equation.

- *An explicit expression of the optimal control u^* in function of y would provide a feedback law which is required for closed-loop control. Such law is derived in the LQ case in a* *subsequent section.*

- *In the case* $g(y(T)) = \frac{1}{2}\|y(T)\|_Q^2$*, we have:* $p^T(T) = -Qy(T)$.
- *The model equation with the I.C. and the adjoint equation with the DF.C. form a "two-point boundary value problem", which can be numerically solved to obtain the solution* u^* *even in the non linear case. This is the basic principle of the indirect methods which are presented in a next section.*

Proof of the theorem. The proof of the theorem is based on *calculus of variations.*
The calculations are similar to those proving the general Theorem 11.15 tackling non-linear stationary PDEs.
It has been proved that the cost function $j(u)$ is strictly convex and the optimal control u^* exists and is unique, see Theorem 10.2.

Let δu be a perturbation to u^*. We denote by u_δ the perturbed optimal control, $u_\delta = (u^* + \delta u)$, and by y_δ the corresponding perturbed solution.

*) y_δ starts from the same I.C. y_0 as y, that is: $\delta u(0) = 0 = \delta y(0)$.

*) We denote by δy the difference $(y_\delta - y^u)$: $y_\delta = y^u + \delta y$.
The perturbed solution satisfies: $y_\delta'(t) = A(t)y_\delta(t) + B(t)u_\delta(t) + S(t)$.
By linearity of the equation, we obtain: $\delta y'(t) = A(t)\delta y(t) + B(t)\delta u(t)$.
Therefore the expression of the solution perturbation:

$$\delta y(t) = M(t)\int_0^t M(s)^{-1}B(s)\delta u(s)ds \tag{10.23}$$

with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t),\ M(0) = Id$.
Recall that if $A(t) = A$ constant then $M(t) = \exp(tA)$.

*) Recall that: $j(u_\delta) = \frac{1}{2}\int_0^T \|y_\delta(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u_\delta(t)\|_U^2 dt + g(y_\delta(T))$.

The optimal control u^* is uniquely determined by the Euler condition $\nabla j(u) = 0$, with:

$$\nabla j(u)\cdot\delta u = \int_0^T < Wy(t), \delta y(t) > dt + \int_0^T < Uu(t), \delta u(t) > dt + \nabla g(y(T))\ \delta y(T) \tag{10.24}$$

(Recall that W and U are symmetric).
The relation $\nabla j(u)\cdot\delta u = 0$ provides the following relation of u in function of y and δy: for all $\delta u(t)$,

$$\int_0^T < \underbrace{Uu(t)}_{\text{sough term}}, \delta u(t) > dt = \underbrace{-\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)...} - \underbrace{\nabla g(y(T)), \delta y(T) >}_{\text{final time term}} \tag{10.25}$$

*) Let us inject the expression of $\delta y(t)$ in function of $\delta u(t)$, see (10.23), in the term in $\delta y(t)$ above, see (10.25). We obtain:

$$\begin{aligned} \underbrace{\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)\dots} &= \int_0^T < WM(t) \int_0^t M(s)^{-1}B(s)\delta u(s)ds, y(t) > dt \qquad (10.26) \\ &= \int_0^T < WM(r) \int_0^T M(s)^{-1}B(s)\delta u(s)ds, y(r) > dr \\ &\quad - \int_0^T \int_0^t y^T(s)WM(s)ds \; M(t)^{-1}B(t)\delta u(t)dt \qquad (10.27) \end{aligned}$$

after integration by parts.

*) By construction, the expression of the adjoint $p(t)$, solution of (10.21) with the F.C. $p^T(T) = -\nabla g(y(T))$, reads:

$$p(t) = \Lambda(T)M^{-1}(t) + \left(\int_0^t y^T(s)WM(s)ds\right) M^{-1}(t)$$

with $\Lambda(T) = -\ \nabla^T g(y(T))M(T) - \int_0^T WM(s)y(s)ds$,
with $M(t) \in M_{n,n}(\mathbb{R})$ such that: $M'(t) = A(t)M(t),\ \ M(0) = Id$.
Recall that if $A(t) = A$ constant then $M(t) = \exp(tA)$.
Note that in the expression of $p(t)$ above, the F.C. $p^T(T) = -\nabla g(y(T))$ is of course retrieved.

By injecting this expression of $p(t)$ (and $\Lambda(t)$) in (10.27), we get:

$$\underbrace{\int_0^T < W\delta y(t), y(t) > dt}_{\text{term in } \delta y(t)\dots} \ = \ etc \qquad (10.28)$$

*) Let us address now on the final time term in (10.25).
By construction of the F.C. of the adjoint field and using the expression of $\delta y(t)$ in function of $\delta u(t)$, see (10.23), at time $t = T$, we get:

$$\underbrace{\nabla g(y(T)), \delta y(T) >}_{\text{final time term}} = etc$$

ToDo: terminer les calculs...

*) By combining the expressions above, we finally obtain the relation: for all $\delta u(t)$,

$$\int_0^T < \underbrace{Uu(t)}_{\text{sough term}} , \delta u(t) > dt = \int_0^T < B^T(t)\ p^T(t), \delta u(t) > dt \qquad (10.29)$$

Therefore the expression:

$$u(t) = U^{-1}(t)B^T(t)\ p^T(t)$$

□

Remark 10.7. *In case of an infinite time interval ($T = +\infty$), the final condition becomes:* $\lim\limits_{+\infty} p(t) = 0.$

10.5.2 The Hamiltonian

W. Hamilton, 1805- 1865, Irish physicist, astronomer and mathematician.
In this section, the central concept of Hamiltonian is introduced. In classical mechanics, the state of a system is described by its generalized coordinates (say q) and their conjugate momenta (say p). The Hamiltonian function $H(q,p)(t)$ is defined as the total energy of the system, that is the kinetic energy plus the potential energy, see e.g.the on-line course [19] on this topic.

Let us consider the linear direct model without source term for sake of simplicity ($S = 0$):

$$\boxed{\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) \text{ for } t \in (0,T) \\ \text{with the Initial Condition: } y(0) = y_0. \end{cases}} \tag{10.30}$$

and the cost function expression:

$$\boxed{j(u(t)) = \frac{1}{2}\int_0^T \left(\|y^u(t)\|_W^2 + \|u(t)\|_U^2\right) dt + g(y(T))} \tag{10.31}$$

In this optimal control context, the *Hamiltonian* H is the functional defined as:

$$\boxed{H(y,p,u)(t) = (p^T, (Ay + Bu))(t) - \frac{1}{2}(\|y(t)\|_W^2 + \|u(t)\|_U^2)} \tag{10.32}$$

with $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$.

The Hamiltonian combines the objective function and the state equation like the Lagrangian in static optimization problems with constraints. Here the multipliers $p(t)$ is a function of time rather than a constant.
Note that the term $g(y(T))$ appears in the F.C. of the adjoint $p(t)$ and not in the Hamiltonian.

Let us calculate the partial derivatives of $H(y,p,u)$. We have:

$$\begin{cases} \partial_y H(y,p,u)(t) \cdot \delta y(t) = (p^T, A\delta y)(t) - (Wy, \delta y)(t) \\ \partial_p H(y,p,u)(t) \cdot \delta p(t) = ((Ay + Bu), \delta p^T)(t) \\ \partial_u H(y,p,u) \cdot \delta u(t) = (p^T, B\delta u)(t) - (Uu, \delta u)(t), \end{cases} \tag{10.33}$$

where $(\cdot,\cdot)$ denotes the scalar product in the adequate Euclidian spaces.

Consequently, the necessary and sufficient conditions of the LQ problem solution stated in Theorem 10.5 correspond to *stationary conditions of the Hamiltonian* in the following sense. For all t,

$$\boxed{\left\{\begin{array}{rcll} y'(t) & = & \partial_p H(y,p,u)(t) & = A(t)y(t) + B(t)u(t) \\ -p'(t) & = & \partial_y H(y,p,u)(t) & = p(t)A(t) - y^T(t)W \\ 0 & = & \partial_u H(y,p,u)(t) & \Leftrightarrow Uu(t) = B^T(t)p^T(t) \end{array}\right.} \tag{10.34}$$

with the Final Condition (F.C.): $\boxed{p(T) = -\nabla g^T(y(T))}$.
(Recall that p is a line vector in $\mathbb{R}^n$).

These three relations constitute the so-called *optimality system.*
The first two equations are the state equation and the adjoint state equations respectively (with the F.C. depending on $y(T)$): they constitute the so-called *Hamiltonian equations.*
The Hamiltonian equations are accompanied by the last equation which is the optimality condition on u, a necessary and sufficient condition in the present LQ case.
In general, these three equations are fully coupled.

The Hamiltonian: the conserved quantity in time

Exercise 10.8. *Let (y^*, p^*, u^*) be the solution of the LQ problem (10.30).*
Show that the mapping $t \mapsto H(y^*, p^*, u^*)(t)$ is constant.

Correction. For all t,

$$\begin{aligned} d_t H(y,p,u)(t) &= \partial_y H(y,p,u)(t)y'(t) + \partial_p H(y,p,u)(t)p'(t) + \partial_u H(y,p,u)(t)u'(t) \\ &= -p'(t)y'(t) + y'(t)p'(t) + 0 \text{ for any solution of the optimality system} \\ &= 0 \end{aligned}$$

Hence the result. □

In some contexts, the Hamiltonian denotes the energy of the system.

Remark 10.9.
- *The solution of the LQ problem exists and is unique, see Theorem 10.2. Moreover, the stationary conditions of the Hamiltonian $H(y,p,u)$ defined by (10.32) corresponds to the necessary and sufficient conditions of the LQ problem solution which is unique, see Theorem 10.5 and Eq. (11.40).*

If the model is non linear or if the objective function $J(u;y)$ is not quadratic then Theorem 10.5 does not hold anymore.
However, in this case, the stationary conditions of the Hamiltonian $H(y,p,u)$ corresponds to necessary conditions of solution(s). In this case, the corresponding triplet(s) (y,p,u) represent optimal control policies for the dynamical system.

- *Some links can be done between the Hamiltonian and the Lagrangian.*
  For the control of PDEs (see next chapter), to obtain the adjoint equation, a Lagrangian will be introduced.

The reader may consult e.g. the on-line course [19] presenting the concepts of "Hamiltonian mechanics" and "Lagrangian mechanics".

10.5.3 Examples & exercises

Exercise 10.10. *Write the Hamiltonian and the resulting optimality system for the optimal control problem applied to the vehicle dynamics presented in Section 10.1.*

For other exercices, please consult the supplementary material.

10.6 Closed-loop control: feedback law and the Riccati equation (LQ case) *

This section is a "to go further section".
Riccati familly (father and son), Italian mathematicians, 18th century.
The optimality condition derived above gives an expression of the optimal control $u(t)$ in function of the adjoint solution $p(t)$. In very simple cases like in some exercises above, it is possible to explicitly integrate the adjoint equation, resulting to an expression of $u(t)$ depending on the state $y(t)$: this states the so-called *feedback law or closed-loop control.* This is the expected information in automation.
However, in general, the expression of u in function of y is far to be trivial... In the LQ case, the feedback law is known: it is obtained by solving the Riccati equation presented below.
Such feedback laws are not required in a Data Assimilation context. Therefore the result below may be skipped for readers interested in DA only.

10.6.1 Feedback law in the LQ case

In the LQ case we have the following result.

Theorem 10.11. *Under the assumptions of the existence - uniqueness Theorem 10.2, the (unique) optimal control u^* writes as a feedback function (closed-loop control) as:*

$$u^*(t) = K(t)y(t) \text{ with } K(t) = U^{-1}B^T(t)E(t)$$

where the matrix $E(t)$, $E(t) \in M_n(\mathbb{R})$, is solution of the Riccati equation:

$$E'(t) = W - A^T(t)E(t) - E(t)A(t) - E(t)B(t)U^{-1}B^T(t)E(t) \quad \forall t \in (0, T)$$

with the Final Condition: $E(T) = -Q$.

For all t, the matrix $E(t)$ is symmetric. Furthermore, since Q and W are positive definite, $E(t)$ is positive definite too.

We refer to [39] for the proof. □
This result provides an expression of the optimal control u in function of the state y.
This enables to build up closed-loop control (vs open-loop control only), see Fig. 10.7.

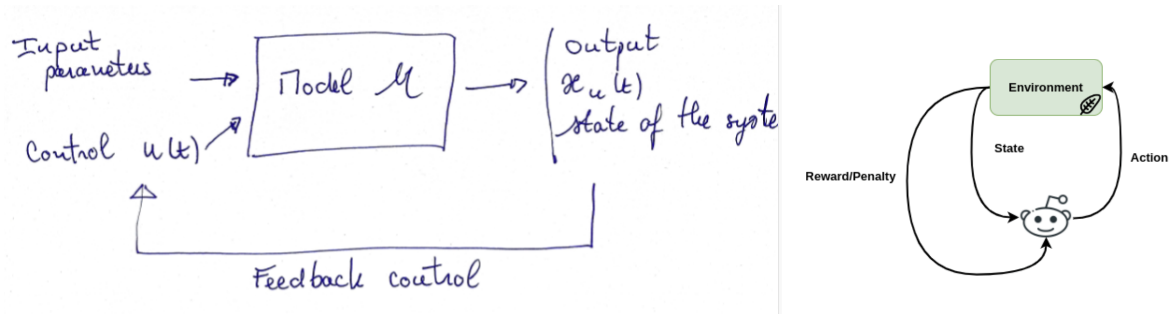


Figure 10.7: (L) Optimal control of a system: closed loop. (R) Reinforcement Learning framework. Image extracted from

10.6.2 Optimal control vs reinforcement learning in machine learning

The aim of optimal control is to determine the best possible policy by minimizing a cost function over time. It is a foundational concept in automation, widely applied in fields such as aeronautics, robotics, and others.

Reinforcement Learning (RL) is a mathematical framework for decision-making that was first introduced in the 1950s.
It has since become a key area of research and application within the broader field of Machine Learning (ML).

There are notable connections between RL and optimal control, as both aim to learn an optimal action policy.

RL is based on stochastic processes (specifically Markov Decision Processes) which are discrete-time optimal control problems involving uncertainty and randomness.

However, there are important distinctions between the two approaches.

Optimal control typically requires complete knowledge of the system's dynamics, while in RL, an agent learns to make decisions by interacting with its environment.
The goal in RL is to maximize the cumulative ”reward” obtained over time. If the agent makes a suboptimal decision, it may incur penalties.
This process in RL involves exploring the environment, taking actions, receiving feedback in the form of rewards, and using this feedback to improve future decision-making.

Optimal control assumes a known model of the system.
On contrary, RL can be applied in situations where the system model is either partially or entirely unknown.

10.6.3 Towards non-linear cases

In practice, optimal control problems are often non-linear. Therefore, the results presented for the LQ problem do not apply directly, especially the analytic expression of the optimal control. Nevertheless, a good understanding of the LQ solution structure, particularly the strict convexity of the resulting cost function, is useful for tackling non-linear or non-strictly convex problems.

Non-linear cases: the Hamilton-Jacobi-Bellman equation

To numerically solve a *non-linear optimal control problem* in the sense of computing an open-loop optimal control u (and the corresponding optimal trajectory y^u), we can follow the same approaches as in the LQ problem: either a *direct method* (for low dimensional problems) or an *indirect method* based on the Pontryagin maximum principle.

To compute an optimal feedback law, we have to consider the so-called *Hamilton-Jacobi-Bellman approach*, see e.g. [19, 39, 7] and references therein, on this wide and complex scientific field.

Connection between the control of ODEs and PDEs

For PDE systems, the Pontryagin principle does not directly apply, and the feedback laws are generally not known. These topics are covered in more detail in the following chapter. This area is still an active field of mathematical research, with recent results including Riccati-like equations for the Navier-Stokes fluid flow equations at low Reynolds numbers, among others.

Despite these challenges, it remains possible to write equations characterizing the optimal control solution for PDE systems. This is achieved through the optimality system based on the adjoint equations, similar to the optimality system derived using the Hamiltonian. This approach is developed further in the next chapter for non-linear elliptic PDE systems.

10.7 Indirect methods (based on the Pontryagin principle) *

This section is a "to go further section".

The Pontryagin principle states that the unique optimal control u^* satisfies the optimality system, as (11.40) in the LQ case.
We present here the numerical approach based on the optimality system. This approach leads to the so-called "Two Points methods". This consists to solve a Boundary Value Problem (BVP) by e.g. the Newton-Raphson algorithm.

The basic principles of this approach are valid whatever if the optimal control is linear or not.

10.7.1 The Boundary Value Problem

By setting $z(t) = (y(t), p(t))$, the two first equation of the optimality system (the state and the adjoint equations) can be written as a first order dynamical system :

$$z'(t) = F(u(t); z(t))$$

with the I.C. $z(0) = (y_0, \cdot)$ and the Final Condition (F.C.) $z(T) = (\cdot, p_T)$.
We write the IC and the FC as: $R(z(0), z(T)) = 0$. (For more details, see e.g. [39]).
Then, the problem to solve reads:

$$\boxed{\left\{\begin{array}{r} z'(t) = F(u(t); z(t)) \\ R(z(0), z(T)) = 0 \end{array}\right.} \tag{10.35}$$

This is a *Boundary Value Problem* (BVP) and not simply a Cauchy problem.

10.7.2 Resulting numerical method

Basic principle Let us denote by $z(z_0; t)$ the solution of the Cauchy problem: $z'(t) = F(u(t); z(t))$, $z(0) = z_0$.
We set: $G(z_0) = R(z_0, z(z_0; T))$. Then, Problem (10.35) consists to solve:

$$G(z_0) = 0 \tag{10.36}$$

Computing the roots of $G(\cdot)$ or solving the BVP (10.35) is equivalent. This problem can be numerically solved by the Newton-Raphson method (under C^2 regularity conditions).

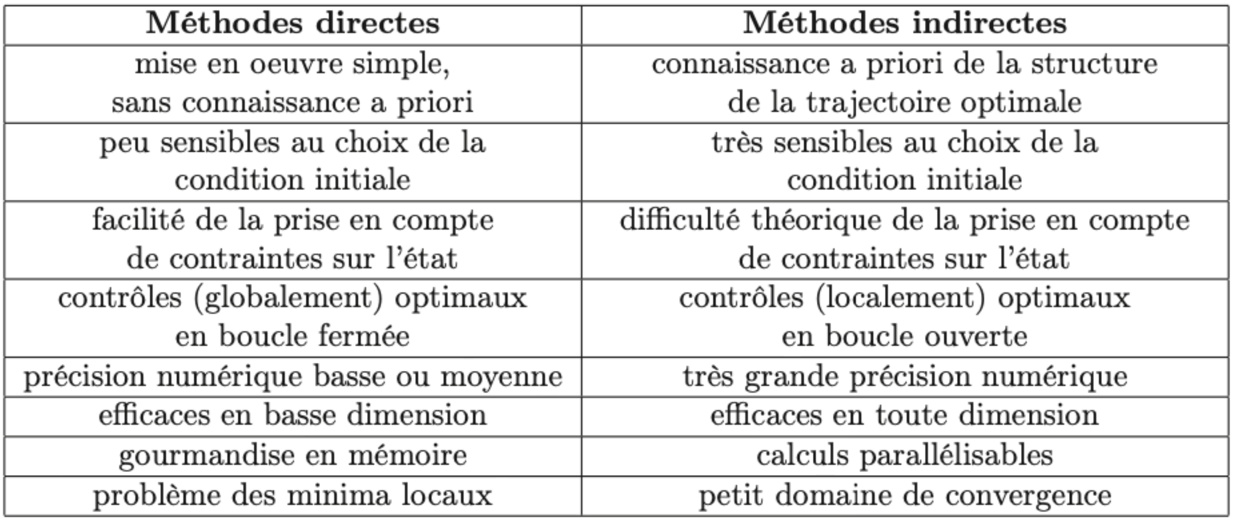
Note that solving 1D BVP like (10.35) by solving successive Cauchy problems is called "shooting methods".

The multiple step version An efficient version of the "shooting method" above consists to split the time interval $[0, T]$ into sub-intervalls $[t_p, t_{p+1}]$ and compute the values $z(t_p)$ at the start of each sub-intervall. Conditions of continuity at each sub-intervall boundary have to be imposed.
The multiple shooting method is more stable than the basic version one.

The reader may refer to [39] and references therein for more details on shooting type methods.

10.7.3 Direct vs indirect methods

Let us briefly present the pros and cons of direct methods presented in Section 10.1 and indirect methods. For details, the author may refer to e.g. [39] and references therein.



Méthodes directes	**Méthodes indirectes**
mise en oeuvre simple, sans connaissance a priori	connaissance a priori de la structure de la trajectoire optimale
peu sensibles au choix de la condition initiale	très sensibles au choix de la condition initiale
facilité de la prise en compte de contraintes sur l'état	difficulté théorique de la prise en compte de contraintes sur l'état
contrôles (globalement) optimaux en boucle fermée	contrôles (localement) optimaux en boucle ouverte
précision numérique basse ou moyenne	très grande précision numérique
efficaces en basse dimension	efficaces en toute dimension
gourmandise en mémoire	calculs parallélisables
problème des minima locaux	petit domaine de convergence

Figure 10.8: Pros and cons of direct methods vs indirects methods (shooting methods) to numerically solve an optimal control problem. Extracted from [39]

10.8 The fundamental equations at a glance (open-loop control)

The Linear model

The linear model (without source term $s(t)$) reads:

$$\begin{cases} \text{Given } u(t), \text{ find } y(t) \text{ such that:} \\ y'(t) = A(t)y(t) + B(t)u(t) \text{ for } t \in I = [0, T] \\ \text{with the initial condition: } y(0) = y_0. \end{cases} \tag{10.37}$$

The model operator (control-to-state map) $\mathcal{M}(u)$ is defined as: $\mathcal{M}(u) = y^u$.
This operator $\mathcal{M}(u(t))$ is here affine in $u(t)$, for all t in $[0, T]$.

The cost functional $j(u)$ is defined from quadratic terms. First, the observation function is defined:

$$j(u(t)) = J(u(t), y^u(t)) = \frac{1}{2}\int_0^T \|y^u(t)\|_W^2 dt + \frac{1}{2}\int_0^T \|u(t)\|_U^2 dt + \frac{1}{2}\|y^u(T)\|_Q^2 \tag{10.38}$$

Given the observation" functional $J(u; y)$, the cost function is defined as:

$$j(u) = J(u\,; y^u) \tag{10.39}$$

with $y^u(t)$ the unique solution of (10.37), given u.

The LQ optimal control problem
Given the I.C. y_0 and the final time T, find $u^*(t)$ such that:

$$j(u^*) = \min_u j(u) \tag{10.40}$$

The Hamiltonian is the conserved quantity in time. Its expressions is:

$$H(y, p, u) = p(Ay + Bu) - \frac{1}{2}(\|y\|_W^2 + \|u\|_U^2) \tag{10.41}$$

with $H : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $u(t)$ the control, $y(t)$ the state of the system and $p(t)$ the adjoint state, solution of the adjoint model.

The adjoint model reads:

$$-p'(t) = p(t)A(t) - y(t)^T W(t) \text{ for } t \in [T, 0] \tag{10.42}$$

with the final condition: $p^T(T) = -Qy(T)$. (p is a line vector).

The Pontryagin maximum principle states that if the control $u(t)$ is defined as:

$$u(t) = U(t)^{-1} B(t)^T p(t)^T \text{ for almost } t \in [0, T] \tag{10.43}$$

then the state $y^u(t)$ associated to this control $u(t)$, is optimal for the LQ problem.

The Pontryagin maximum principle can be read as follows: the equations below have to be satisfied.

$$\left\{ \begin{array}{rcll} y'(t) & = & \partial_p H(y, p, u) & = Ay(t) + Bu(t) \\ -p'(t) & = & \partial_y H(y, p, u) & = p(t)A - y^T(t)W \\ 0 & = & \partial_u H(y, p, u) & \Longleftrightarrow Uu(t) = B^T p^T(t) \end{array} \right. \tag{10.44}$$

with the final condition: $p(T) = -\nabla^T g(y(T)) = -y(T)^T Q$. This is the **optimality system**.

Chapter 11

Optimal Control of Stationary PDEs: Adjoint Method, VDA

This chapter presents optimal control basis for PDE systems. This leads to Variational Data Assimilation (VDA) formulations. The final goal is the computational algorithms to estimate control variable even if they are large dimension. The algorithms rely on the derivation of the first order optimality system which is based on the adjoint equations.
The calculations are derived for a general non-linear elliptic PDE model (stationary) therefore formerly valid for a large class of models.
Firstly, a brief presentation of the equations is provided in discrete dimension for readers who are not confortable with infinite dimensions, functional analysis, and differential calculus. Secondly the equations in infinite dimensions (in Hilbert spaces) are derived more rigorously. Theorem 11.38 states a general result which can be applied to any stationary PDE to obtain the adjoint equation and the cost gradient of a particular problem.

The resulting computational control algorithm leading to VDA is highlighted. It is known as the 3D-Var algorithm in the DA literature. This algorithm enables the fitting of model outputs to data, allowing for the *identification of uncertain input parameters* and *model calibration.* Examples are presented.

Some results and developments require relatively high mathematical skills; this is particularly the case when addressing the differentiability of the PDE solution (with respect to the control variable) and when addressing the existence and uniqueness of the optimal control in the LQ case. These mathematical developments are gathered in a dedicated section entitled "mathematical purposes". This section can be skipped by readers interested in practicals and algorithms only.
Before reading this section, the reader may need to revise basic concepts in functional analysis. Some of them are recalled in Appendix.

It is supposed here that the reader is aware of basic optimization concepts and gradient-based minimization algorithms. A few of theses basic concepts are shortly recalled in Appendix as well (including a few exercises extracted from the literature).

For a deeper exploration of the topic of optimal control of PDEs (not necessarily the LQ problem and without addressing VDA), the reader may refer to, e.g. [41].

The outline of this chapter is as follows[1].

Contents

[1] Recall that the sections indicated with a * are ”to go further sections”. They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

11.1 General non-linear case in infinite dimension

Let us go back to infinite dimensional spaces and general non linear elliptic type BVP.

11.1.1 The direct model

Let Ω be a bounded domain (Ω Lipschitz). Let U be the controls space. U is supposed to be a Hilbert space.
U Banach space only is potentially enough, however for simplicity it is here considered as a Hilbert space.
Let V be the states space, V a Hilbert space.

We consider the following general direct model (the state equation):

$$\boxed{\begin{cases} \text{Given } u \in U, \text{ find } y \in V \text{ such that:} \\ A(u;y) = F(u) \text{ in } \Omega \\ \text{with BCs(u) on } \partial\Omega \end{cases}} \tag{11.1}$$

where A is an elliptic operator (with respect to the state y), defined from $U \times V$ into V' (dual of V).
$A(\cdot;\cdot)$ is a-priori non-linear, both with respect to the parameter u and with respect to the state y.
F is defined from U into V'.

Assumption 11.1. *The state equation (11.1) is well posed in the Hadamard sense: it has an unique solution $y \in V$, moreover this solution is continuous with respect to the parameters (in particular with respect to u).*

In the linear case (A elliptic operator linear with respect to y) the Lax-Milgram theorem might be the right framework to prove the existence-uniqueness, while the Mint-Browner theorem is useful for a large class of non-linear cases, see e.g. [22].

Optimal control terminology: distributed control, boundary control

- If the control u appears in the "bulk" (i.e. in Ω) then one says that it is a *distributed control.*
- If u appears on the boundary conditions only, then one says that it is a *boundary control.*

11.1.2 Examples

Two simple linear examples based on second order elliptic operators are as follows.

Example 1)

$$A(u;y) = -div(\lambda \nabla y) \text{ and } F(u) = u$$

with mixed boundary conditions: $y = 0$ on Γ_0 ; $-\lambda \partial_n y = \varphi$ on $\partial\Omega/\Gamma_0$, with φ given.
$\lambda \in L^\infty(\Omega)$, $\lambda > 0$ a.e.
The corresponding functional spaces are : $U = L^2(\Omega)$, $V = H^1_{\Gamma_0}(\Omega)$.
The control u constitutes the source term (the RHS). It represents an external force (since in the RHS).
It is a (spatially) distributed control.
The state equation models a diffusion phenomena e.g. heat diffusion in a structure, concentration in a fluid, the elastic deformation of a membrane under external force $f = u$, the electrostatic field in a conducting media, etc.

Exercice 11.2.
a) Write the state equation (and recall the adequate functional spaces).
Prove that it has one and only one (weak) solution in V.
b) Prove that the unique solution y is continuous with respect to u
i.e. the operator $\pi : u \in U \mapsto y^u \in V$ is continuous.

Correction.
a) In vertu of Lax-Milgram theorem.
b) The inequalities of continuity and coercivity give the result. □

Example 2)

A non linear version of the previous example is as follows.

$$A(u; y) = -div(\lambda(y; u)\nabla y)$$

with mixed boundary conditions (independent of u).
In this still classical case, the PDE is non linear due to the term $\lambda(y; \cdot)$.

Example 3)

Let us consider: $A(u; y) = -div(\lambda \nabla y)$ and $F(u) = f$, with (λ, f) given in adequate functional spaces, and the BC defined by:

$$y = 0 \text{ on } \Gamma_0 \text{ and } -\lambda \partial_n y = u \text{ on } \partial\Omega/\Gamma_0$$

In this case, u is a boundary control representing a flux at the boundary.

The correct functional spaces are : $U = L^2(\Omega)$, $V = H^1_{\Gamma_0}(\Omega)$.
As previously, the corresponding state equation has one and only solution in V (in vertu of Lax-Milgram theorem). The inequalities of continuity and coercivity show the continuity of y with respect to u.

11.1.3 The objective function terms

The formalism is the same as previously presented.
One seek to make fit "at best" (in the least-square sense here) the model outputs to data. The observation operator Z mapping the state of the system y onto the observation space $\mathcal{O}$ is introduced as : $Z : y \in V \mapsto z \in \mathcal{O}$ with $\mathcal{O}$ here supposed to be a Hilbert space. The observation operator Z is a-priori non-linear.

Z may represent a complex non-linear multi-scale model e.g. a map between sequences of 2D optical images representing a 3D fluid dynamic flow to 3D velocity fields.

Next, a natural definition of the observation function is as already presented as:

$$J_{obs}(y) = \|Z(y) - z^{obs}\|^2_{R^{-1}} \tag{11.2}$$

The misfit is measured in the observation space $\mathcal{Z}$, using the norm $\|\cdot\|_R$. The operator R^{-1} is symmetrical positive (semi-)definite therefore defining (semi-)norms.
Recall that in the Linear-Quadratic-Gaussian case, the optimal definition of R^{-1} relies on a covariance matrix of the observation errors R (see Section **??**).
In practice, because of lack of information, R is often simply diagonal, the diagonal coefficients represent the a-priori confidence we have on each observation.
If enriched by a regularization term, the objective function reads as:

$$\boxed{J(u\,;y) \;=\; J_{obs}(y) \;+\alpha_{reg}\; J_{reg}(u)} \tag{11.3}$$

with J_{reg} the regularization term.
Note that $J : U \times V \to \mathbb{R}$ with $J_{obs} : V \to \mathbb{R}$ and $J_{reg} : U \to \mathbb{R}$.

The cost function $j(u)$ is finally defined as:

$$\boxed{\begin{array}{c} j(u) = J(u\,;y(u)) \\ \text{where } y(u) \text{ (also denoted } y^u\text{) is the (unique) solution of the direct model (11.1).} \end{array}} \tag{11.4}$$

We have: $j : U \to \mathbb{R}$.

Classical regularization terms

Classical regularization terms are as follows.
a) If a "first guess"/ "background" value u_b is provided (value which is supposed to a good first estimation of the unknown parameter u), then one may consider the term:

$$J_{reg}(u) = \left\| u - u_b \right\|^2_{B^{-1}} \tag{11.5}$$

with B^{-1} symmetrical positive (semi-)definite, defined as discussed in Section **??**.

This additional term $J_{reg}(u)$ is quadratic (in u) therefore strictly convex...

b) If one seeks to impose higher regularity on the parameter u, one may set:

$$J_{reg}(u) = \left\| \nabla u \right\|_2^2 \text{ or even } \left\| \Delta u \right\|_2^2 \tag{11.6}$$

This term enforces to find the optimal solution u with higher regularity therefore in a smaller sub-space.

Recall that $\int \|\nabla u\|^2 dx$ corresponds to the energy of the Laplace operator Δu (isotropic linear elastic model solution). $\int \|\Delta u\|^2 dx$ corresponds to the energy of the bi-Laplacian operator $\Delta^2 u$ (plate model solution).

Note that $J_{reg}(u)$ is here not quadratic in u anymore...

More sophisticated regularization term expressions may be derived from physical or probabilistic analyses. This point is discussed in a next chapter.

LQ problems vs real-world problems Most of the control problems are not LQ problems for one of at least one of the following reason:

- The model is not linear.
- The regularization term $\|J_{reg}(u)\|$ is higher order only (e.g. of the form $\|\nabla u\|^2$) therefore not strictly convex in u (but in $\|\nabla u\|$ only).
- Even if the direct model is linear then we generally do not have measurements everywhere in Ω, that is at each numerical grid points/nodes!

In real-world problems, data z^{obs} are generally available at some locations only or densely but at larger scale compared to the model resolution scale. For example, satellite observations are generally too large scale (therefore averaged observations) to measure small scale dynamics. Another example are camera measurements of a material presenting complex multi-scale structures. As a consequence, the actual misfit term may have one of the following form:

$$\int_\omega (Z(y) - z^{obs})^2 \ dx \text{ or } \sum_{m=1}^{M} (Z(y)(x_m) - z^{obs}(x_m))^2 \text{ or } \int_\Omega (Z(y) - \mathcal{F}(z^{obs}))^2 \ dx$$

where ω is a non-convex subset of the complete domain Ω, M is the total number of point-wise data and $\mathcal{F}$ denotes e.g. an uncertain low-pass band filter.

In one of these cases, the cost function is not strictly convex anymore and the uniqueness of a minimum u^* is not guaranteed anymore.

11.1.4 Variational Data Assimilation (VDA) is based on the optimal control of the model

Variational Data Assimilation (VDA) is based on the optimal control of the model (here a PDE), where the cost function $j(u)$ quantifies the discrepancy between the observed data and the model's output(s).

Let U_{ad}, subset of U, be the admissible control set.The optimal control problem reads:

$$\boxed{\begin{cases} \text{Find } u^* \in U_{ad} \text{ such that:} \\ j(u^*) = \min\limits_{U_{ad}} j(u) \\ \quad \text{with } j(u) = J(u\,; y(u)), \text{ where } y(u) \text{ solution of the direct model, see (11.4).} \end{cases}} \tag{11.7}$$

In the cost function expression, $y^u \equiv y(u)$ depends on u through the control-to-state mapping $\mathcal{M}$, $\mathcal{M} : u \in U \mapsto y^u \in V$, y^u the (unique) solution of the direct model (11.1).

Problem (11.7) can be re-read as:

$$\boxed{\begin{cases} \text{Minimize } j(u) = J(u; y^u) \text{ in } U_{ad} \\ \quad \text{under the "model constraint" (11.1)} \end{cases}} \tag{11.8}$$

The formulation (11.8) shows that the problem is an optimization problem under the constraint "the model is satisfied".
This point of view naturally leads to introduce the Lagrangian of this optimization problem. This is the approach followed in next section.

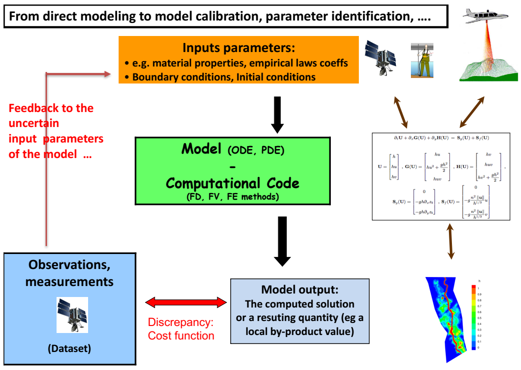

From direct modeling to model calibration, parameter identification, ...
Inputs parameters:
• e.g. material properties, empirical laws coeffs
• Boundary conditions, Initial conditions
Feedback to the uncertain input parameters of the model ...
Model (ODE, PDE)
-
Computational Code
(FD, FV, FE methods)
Observations, measurements
(Dataset)
Discrepancy: Cost function
Model output:
The computed solution or a resuting quantity (eg a local by-product value)
$\partial_t U + \partial_x G(U) + \partial_y H(U) = S_g(U) + S_f(U)$

Figure 11.1: Principle of VDA: Control the uncertain input parameters u of the model to fit the model output y to the data z^{obs}.

11.1.5 Why computing the gradient $\nabla j(u)$?

Computing the gradient is obviously required first step before employing gradient-based optimization algorithms.

If the dimension of the (discrete) parameter u is small, say $\mathcal{O}(1)$, and if the evaluation of $j(u)$ can be done in extremely short time, say in ms, then the optimization problem (11.8) can be solved by employing global gradient-free methods, such as Monte-Carlo - based algorithms or genetic algorithms for example.

On contrary, if the dimension of the (discrete) parameter u is large, moreover if the evaluation of $j(u)$ is CPU-time consuming (this is generally the case if considering PDE models), then the optimization problem has to be solved by descent algorithms aiming at computing a local minimum only.
And whatever the descent algorithms (L-BFGS etc) require the information of the gradient $\nabla j(u)$.

The computation of the gradient $\nabla j(u)$ in large dimensional cases (u is of large dimension) can be not straightforward to compute.

Different ways to compute the gradient $\nabla j(u)$ are discussed in next sections, including the one based on the adjoint equations.

11.2 A few mathematical foundations

11.2.1 Differential calculus in functional spaces (infinite dimensions)

Please consult the Appendices and the supplementary material.

11.2.2 Weak forms and dual space representation

Considering weak forms of the equations is interesting in mathematical analysis e.g. aiming at characterizing "weak" solutions that is solutions in larger functional spaces then the $C^k(\bar{\Omega})$ spaces ($k = 2$ for 2nd order PDEs for example). Weak solutions enable to represent observed (real-world) singularities such as discontinuities, shocks etc. Also, weak formulations constitute the first key stage to elaborate Finite Element schemes.
In the presence of non trivial boundary conditions, which is often the case in real-world problems, considering the weak form of the equations enables to naturally and rigorously derive the adjoint equations.

The direct model equation in Ω reads: $A(u;y)(x) = F(u)(x)$ for $x \in \Omega$, plus BCs, see (11.1).

For a so-called regular solution y, that is $y \in C^k(\bar{\Omega})$ ($k = 2$ for 2nd order PDEs), the equation is satisfied for all $x \in \Omega$.
On the opposite, the weak solution denotes the solution of the weak formulation of the BVP. Weak formulations are also called variational formulations.

In practice, stating the weak formulation of a BVP consists to:

- Multiply the equation by a test function z which belongs to the same functional space V as y (e.g. the Sobolev space $H^1(\Omega)$ for a second order linear PDE plus the Dirichet BC),
- Integrate over the domain Ω and use integration by parts (Green's formulae) to transfer the highest order derivatives from the primary function y to the test function z. This reduces the differentiability order requirement!
- Apply the homogeneous Dirichlet BC on the test function z: this cancel the boundary integral on the Dirichlet boundary.
  Apply the flux (constraint) type BCs on y.

By setting forms as:

$$a(u; y, z) : U \times V \times V \to \mathbb{R} \text{ and } b(u; z) : U \times V \to \mathbb{R}$$

the weak formulation of the direct model (the state equation) reads:

$$\boxed{\begin{cases} \text{Given } u \in U, \text{ find } y \in V \text{ such that:} \\ a(u; y, z) = b(u; z) \text{ for all } z \in V \end{cases}} \tag{11.9}$$

If the direct model is linear (i.e. with respect to the unknown y) then the form $a(.; y, z)$ is bilinear.
On the contrary, if it is not, then $y \mapsto a(.; y, .)$ is not linear.
However, $z \mapsto a(.; ., z)$ and $z \mapsto b(.; z)$ are necessarily linear. As a consequence, one can set:

$$\boxed{< A(u; y), z >_{V' \times V} = a(u; y, z) \text{ and } < F(u), z >_{V' \times V} = b(u; z)} \tag{11.10}$$

and the classical form of the equation in Ω is satisfied by the weak solution in V':

$$\boxed{A(u; y) = F(u) \text{ in } V'} \tag{11.11}$$

Exercice 11.3. *Apply this presentation to the toy BVP: $-div(u\nabla y) = F$ in Ω with non-homogeneous mixed BCs on $\partial\Omega$.*

Note that for V Hilbert space, the Riez-Frechet representation theorem applies (see Appendix and supplementary material).

11.2.3 Connection between the differential $j'(u)$ and the gradient $\nabla j(u)$

As previously discussed, to numerically solve the optimal control problem with large dimension (discrete) control variable, one needs to employ descent algorithms which are based on the gradient information $\nabla j(u)$.

To compute the gradient one first needs to clarify what is the link between the differential $j'(\cdot)$ and the gradient $\nabla j(\cdot)$.
To perform computations, the state equation (direct model) has to be discretized using an adequate numerical method e.g. finite differences, finite elements, finite volumes.
Recall that $j : U \to \mathbb{R}$, then $j'(u) \in \mathcal{L}(U; \mathbb{R})$.
Let us denote U_h the discrete control space with $dim(U_h) = m$ (there is m discrete control variables). We assume that $U_h \subset U$.

The gradient $\nabla j(u)$ to be computed (with $u \in U_h$), $\nabla j(u) \in \mathbb{R}^m$, is related to the differential $j'(u)$ by the relation:

$$\boxed{< \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m} \tag{11.12}$$

Of course, the functional $j(\cdot)$ is here assumed to be differentiable.
In the sequel, sufficient conditions are presented to have $j(\cdot)$ continuously differentiable that is of class C^1.

Note that a few exercises on differential calculus is proposed in the supplementary material of this course. Also one may do the following exercise.

Exercice 11.4. *Let j be the cost function defined by $j(u) = J(u; y^u)$ with J defined by (11.3).*
a) Write a sufficient condition to have j of class C^1.
b) Write an expression of the differential $j'(u).\delta u$, for all δu
c) Is the cost function $j(u)$ strictly convex ?
Discuss the answer depending on the observation operator Z and the regularization term form.
Recall: linear + quadratic implies strictly convex.

11.3 Equations derivation from the Lagrangian

As already mentioned, the optimization problem (11.8) may be read as a standard differentiable optimization problem with the model (11.1) viewed as an equality constraint.

11.3.1 The Lagrangian

All calculations below are formal: we do not pay attention to functionals spaces. The equations may be read as being discrete systems too.

The optimization problem (11.8) may be viewed as a differentiable optimization problem with the model (11.1) being an equality constraint.
Then, it is natural to write the corresponding Lagrangian $\mathcal{L}$:

$$\boxed{\mathcal{L}(u; y, p) = J(u; y) - < A(u; y) - F(u), p >} \tag{11.13}$$

with p the Lagrangian multiplier.

- If considering that $A(u; y)$ and $F(u)$ denote the PDE terms in finite dimension, e.g. $A(u; y)$ the rigidity matrix in FEM and $F(u)$ the RHS vector, then $< \cdot, \cdot >$ simply denotes the Euclidian scalar product.

- If considering that $A(u; y)$ and $F(u)$ represent the operators of the PDE, e.g. $A(u; y) = -div(u\nabla y)$, then $< \cdot, \cdot >$ denotes the dual product $< \cdot, \cdot >_{V' \times V}$ with V the state y belongs to, V a Hilbert space.
  In this case, p is a dual variable belonging to V too.

No constraint are here imposed to the control u, neither equality nor inequality ones, otherwise additional lagrangian multipliers related to constraints on u would be introduced.

11.3.2 The optimality system

The stationary point(s) of the Lagrangian provide the necessary optimality condition. These points are determined by the relation: $\nabla \mathcal{L}(u; y, p) = 0$. This reads:

$$\boxed{\left\{ \begin{array}{l} \partial_u \mathcal{L}(u; y, p) \cdot \delta u = 0 \quad \forall \delta u \\ \partial_y \mathcal{L}(u; y, p) \cdot \delta y = 0 \quad \forall \delta y \\ \partial_p \mathcal{L}(u; y, p) \cdot \delta p = 0 \quad \forall \delta p \end{array} \right.} \tag{11.14}$$

The last equation of (11.14) provides the direct model: $A(u; y) = F(u)$.

The second equation of (11.14) provides the following linearized equation:

$$\partial_y J(u; y) \cdot \delta y - < \partial_y A(u; y) \cdot \delta y, p >= 0 \quad \forall \delta y$$

Therefore: $< \partial_y J(u; y) - [\partial_y A(u; y)]^* \cdot p, \delta y >= 0 \;\; \forall \delta y$.
Therefore:

$$[\partial_y A(u; y)]^* \cdot p = \partial_y J(u, y)$$

This is the adjoint equation.

The first equation of (11.14) reads: $\partial_u J(u;y)\cdot\delta u - < (\partial_u A(u;y) - F'(u))\cdot\delta u, p >= 0\ \forall\delta u$.

It will be shown later that this equation is the necessary condition which reads: "the gradient equals 0".

Using the particular decomposition of $J(u;y)$ introduced in (11.3), and by setting $\alpha_{reg} = 1$, we obtain:

$$\partial_y J(u,y)\cdot\delta y = J'_{obs}(y)\cdot\delta y \text{ and } \partial_u J(u;y)\cdot\delta u = J'_{reg}(u)\cdot\delta u \tag{11.15}$$

In summary, we have the set of equations:

$$\boxed{\left\{\begin{array}{lrcll} \text{Given } u\text{, find } y \text{ s.t. } : & A(u;y) & = & F(u) & \text{(Direct model)}\\ \text{Given } (u,y)\text{, find } p \text{ s.t. } : & [\partial_y A(u;y)]^*\cdot p & = & J'_{obs}(y) & \text{(Adjoint model)}\\ \text{Given } (y,p)\text{, find } u \text{ s.t. } : & [\partial_u A(u;y) - F'(u)]^*\cdot p & = & J'_{reg}(u) & \text{(1st order condition)} \end{array}\right.} \tag{11.16}$$

This set of equations constitutes the optimality system.

11.3.3 In weak forms

Let V be the state space ($y \in V$), V a Hilbert space. The Lagrangian reads:

$$\begin{array}{rcl} \mathcal{L}(u;y,p) & = & J(u;y) - [a(u;y,p) - b(u;p)] \\ \mathcal{L} & : & U\times V\times V \to \mathbb{R} \end{array}$$

Next, the last equation of (11.14) provides the state equation (the direct model in weak form):

$$a(u;y,\delta p) \ = \ b(u;\delta p) \quad \forall\delta p \in V$$

The second equation of (11.14) provides the following linearized equation:

$$\partial_y a(u;y,p)\cdot\delta y \ = \ \partial_y J(u,y)\cdot\delta y \quad \forall\delta y \in V$$

It is the adjoint equation.

The first equation of (11.14) reads:

$$\partial_u J(u;y)\cdot\delta u - [\partial_u a(u;y,p) - \partial_u b(u;p)]\cdot\delta u = 0 \ \ \forall\delta u \in U$$

It will be shown that: $j'(u)\cdot\delta u = \partial_u J(u;y)\cdot\delta u - [\partial_u a(u;y,p) - \partial_u b(u;p)]\cdot\delta u$.
Therefore this last equation reads: $j'(u) = 0$.

These three equations summarizes as:

$$\left\{\begin{array}{rcll} a(u;y,z) & = & b(u;z) & \forall z \in V \\ \partial_y a(u;y,p)\cdot z & = & \partial_y J(u,y)\cdot z & \forall z \in V \\ \partial_u J(u;y)\cdot \delta u - [\partial_u a(u;y,p) - \partial_u b(u;p)]\cdot \delta u & = & 0 & \forall \delta u \in U \end{array}\right. \tag{11.17}$$

Exercice 11.5. *Apply the general expressions above to the equations of the programming practical.*

11.4 Mathematical purposes *

This is a "to go further section".

11.4.1 Differentiability of the cost function

In the following, we will need to differentiate the cost function j (with respect to its unique variable u). Thus, the following question is of main interest in order to address the optimal control problem:

Is the cost function (continuously) differentiable ?

This question of differentiability is potentially difficult to answer for non-linear systems. For non-linear hyperbolic system in particular, the solution may be even not continuous with respect to the control variable u...

A useful result to address this question of differentiability is the *implicit function theorem.*

Theorem 11.6. *(Implicit function theorem) Let us assume that:*
i) the operator A and F in the state equation (11.1) are C^1 (i.e. $A \in C^1(U \times V)$ and $F \in C^1(U)$),
ii) the linearized problem is well-posed (i.e. given (u_0, y_0) the linearized operator $\partial_u A(u_0; y_0)$ is an isomorphism from V into V').

Then, the operator (the "implicit function") $\mathcal{M} : u \mapsto y^u$, with y^u is the (unique) solution of the state equation, is locally C^1 (locally means it exists a neighborhood of u_0 such that).

In short, in view to apply the implicit function theorem we need to verify that the state operators are C^1 and the linearized problem is well-posed.

Here the "implicit function" is the "model operator" $\mathcal{M}(u)$ defined by (**??**).

Example 3) We set: $A(u;y) = -u\triangle y$, $F(u) = f$, with mixed boundary conditions: $y = 0$ on Γ_0 ; $-u\partial_n y = \varphi$ on $\partial\Omega/\Gamma_0$, with φ given.
Here, u is a distributed control, it is the diffusivity coefficient of the material.

Exercice 11.7.
a) Write the corresponding state equation, and prove that it has we and only (weak) solution in V for u given in $L^\infty(\bar{\Omega})$, $u > 0$ a.e..
b) Prove that the unique solution y is continuous and differentiable with respect to u (in the right functional space).

11.4.2 Existence and uniqueness of the optimal control in the LQ case

Warm up with a basic linear finite dimensional problem

Let us consider a finite dimensional linear direct model, then, the control-to-state map M reads as:

$$M : u \in \mathbb{R}^m \mapsto y^u \in \mathbb{R}^n$$

The following example fits into this problem category:

$$\boxed{Ay = Fu \text{ in } \mathbb{R}^n} \tag{11.18}$$

with $A \in \mathcal{M}_{n\times n}$ a non singular real matrix, F a rectangular matrix, $F \in \mathcal{M}_{n\times m}$, $m < n$, of maximal rank m.

We consider the usual quadratic observation function $J(u;y) = \|Zy - z^{obs}\|_2^2$, with the observation operator Z a non-singular linear matrix Z of $\mathcal{M}_{n\times n}$.
For a sake of simplicity, we set here $z^{obs} = 0$.

The cost function is defined as usual as: $j(u) = J(u;y^u)$.
The optimal control problem aims at solving $\boxed{u^* = arg \min_{u\in R^m} j(u)}$.

Exercise 11.8. *Show that under the maximal rank assumptions above, this optimal control problem admits an unique solution u^*, even without regularization term in $J(u;y)$.*

Correction. We have: $\boxed{y^u = A^{-1}Fu = Mu}$. The control-to-state map M is linear.

Let us set $\boxed{N = (Z \circ M)}$, N non-singular matrix of $\mathcal{M}_{n\times n}$. We have:

$$j(u) =< Nu, Nu >_2=< N^T Nu, u >_2= \|u\|_{N^TN}^2$$

Indeed, the matrix $N^T N$ is symmetric positive definite (since Z and M invertible), therefore defining a norm. Consequently, $j(u)$ is strictly convex: $\forall (u_1, u_2) \in M^2$, $\forall t \in]0,1[$,

$$j(tu_1 + (1-t)u_2) < tj(u_1) + (1-t)j(u_2)$$

unless $u_1 = u_2$. **ToDo: True ?**

Consequently, the optimization problem admits an unique minimum u^*. □

Back to the general continuous case

In the general case (moreover in infinite dimension), the idea remains the same as the previous basic linear case.
Calculations are a bit heavier due to the non vanishing dataset z^{obs}, also due to the more general observation operator Z (however still linear).

Recall that the state equation reads:

$$\boxed{A(u; y) = F(u) \text{ in } V'}$$

with V' the dual space of the Hilbert space V.
Next, the cost function $j(u)$ is defined from the observation function $J(u; y)$, see (11.4), which is defined as the sum of the data misfit term $J_{obs}(y)$ and a regularization term $J_{reg}(u)$, see (11.3).

The state equation operator is said to be coercive in V if for all $y \in V$, for all $u \in U$, there exists $\alpha > 0$ such that:

$$< A(u; y), y >_{V' \times V} = a(u; y, y) \ \geq \ \alpha \|y\|^2$$

The regularization term may have different forms. If $J_{reg}(u)$ is quadratic, therefore strictly convex, as defined by (11.5), this defines a LQ problem.

In the LQ case, the existence and uniqueness of the *optimal control* u^* hold.

Theorem 11.9. *Let us assume that the state equation operator $A(\cdot; y)$ is linear and coercive. Let us consider the observation function $J(u; y)$ defined as (11.3) with the regularization term (11.5). Assume moreover that U_{ad} is a closed convex subset of U.*
Then, it exists a unique solution u^* at the optimal control problem (11.7).

Proof derived from those presented in [32, 31] Chapter 1.
Past Step 0) below, the proof is very similar to those of Theorem 10.2.

Let us consider the expression of $J(u;y)$ as in (11.3)-(11.5) with $u_b = 0$. The cost function satisfies $j(u) = J(u;y^u)$.

Step 0) We first reformulate the cost function expression as follows:

$$j(u) = \|Z(y^u - y^0) + Zy^0 - z^{obs}\|^2 + \|u\|^2_{B^{-1}}$$

with y^0 given in V. We set:

$$\pi(u,v) = (Z(y^u - y^0), Z(y^v - y^0))_Z + (Bu, v) \text{ and } D(v) = (z^{obs} - Zy^0, Z(y^v - y^0))_Z$$

Then, the cost function reads:

$$j(u) = \pi(u,u) - 2D(u) + \|z^{obs} - Zy^0\|^2_Z$$

Since the model operator $\mathcal{M}$ is affine and continuous, the form π is bilinear symmetric in U. Moreover, due to the quadratic term $\|u\|^2_{B^{-1}}$, we have the U-coercivity estimation:

$$\pi(u,u) \geq c_0\|u\|^2 \ , \ c_0 > 0, \ \forall u \in U \tag{11.19}$$

The form $D(u)$ is linear continuous in U. Therefore $j(u)$ is continuous and satisfies:

$$j(u) \geq c_0\|u\|^2 - c_1\|u\| \tag{11.20}$$

for a given $c_1 > 0$.

From now, the proof is very similar to those for ODEs, see Theorem 10.2.

A) Proof of existence. It is based on the convergence of minimizing sequence (calculus of variations, D. Hilbert, 1900 a.c. approx.).

Step 1). Let (u_n) be a minimizing sequence:

$$j(u_n) \rightarrow_n \inf\{j(u), u \in U_{ad}\} \tag{11.21}$$

From (11.20)(11.21), we obtain:

$$\|u_n\| \leq constant$$

Hence there exists a sub-sequence (u_{n_k}) which converges weakly to a control u in U_{ad}:

$$u_{n_k} \rightharpoonup u \text{ in } U_{ad}$$

Step 2). U_{ad} is a closed convex subset hence *weakly* closed. Hence $u \in U_{ad}$.

Step 3). Since the cost function $j(u)$ is continuous (lower semi-continuous would be enough), we have: $j(u) = \min_{v \in U_{ad}} j(v)$. In other words, u is solution of the optimal control problem.

B) Uniqueness. The bilinear form $v \mapsto \pi(v, v)$ is coercitive, see (11.19), therefore the cost function is strictly convex.
The uniqueness of u^* is a straightforward consequence of the strict convexity of $j(u)$. □

Exercice 11.10. *Detail the proof of uniqueness.*

Correction.
Hint: It is very similar to the proof in the ODE case. □

Cases with higher-order regularization terms $J_{reg}(u)$. In the case $J(u; y)$ is decomposed as previously, see (11.3), however with a higher-order regularization term such as e.g. (11.6), $J_{reg}(u)$ is generally not strictly convex anymore... In such a case, the existence holds but the uniqueness may not.

However, if the typical case $U = H^1_\Gamma(\Omega_0)$, $H^1_\Gamma(\Omega_0) = \{v \in H^1(\Omega),\ \ v|_{\Gamma_0} = 0\}$, and in presence of a 1st order regularization term such as $\|\nabla u\|^2$, then in vertu of the Poincaré's inequality, the estimation (11.20) still holds and the proof remains the same.
In this case, the control variable u is controlled by its gradient (the regularization term) plus a given value at some locations (on Γ_0).

11.5 Gradient computation: methods for small dimension cases

11.5.1 Recall: why and how to compute the cost function gradient?

To solve the optimization problem (11.8) few approaches are a-priori possible, global optimization methods or local minimization methods. The choice mainly depends on the CPU time (denoted by T^j_{cpu}) required to evaluate the cost function $j(u)$ and on the control variable dimension m (therefore the dimension of the gradient $\nabla j(u)$).

If T^j_{cpu} is small (let say in fractions of seconds using your laptop or a super-computer, whatever), then one can adopt a *global optimization approach* based on stochastic algorithms e.g. Monte-Carlo type algorithms, heuristic methods (e.g. genetic algorithms) or surface response approximation.

When the state equation is a PDE system, T^j_{cpu} is generally not small enough to do so. In this case, global optimization is not worth considering. Then, one has to adopt *local minimization* approaches based on *algorithms of descent*. Then the computation of the cost function gradient is required.

If the m is large then one very likely needs to employ descent algorithms, therefore to compute the cost function gradient $\nabla j(u_h)$.

Before going further, let us recall the relation ship between differential $j'(u)$ in a given direction δu and the gradient value in this direction, see 11.12: $< \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u$ for all $\delta u \in U_h \subset \mathbb{R}^m$.

Problem statement

In the discrete context, the dimension of the gradient $\nabla j(u)$ equals m, m the dimension of the discrete control variable.
Descent algorithms require scalar products of the form $< \nabla j(u), \delta u >$ for at least m directions δu.

Composite control variable case In the case the control variables includes different natures of components e.g. $u = (u_1, u_2)$ then we have:

$$\nabla j(u) = (\frac{\partial j}{\partial u_1}(u), \frac{\partial j}{\partial u_2}(u))^T$$

$$j'(u) \cdot \delta u = \frac{\partial j}{\partial u_1}(u) \cdot \delta u_1 + \frac{\partial j}{\partial u_2}(u) \cdot \delta u_2$$

Small dimensional vs large dimensional case In practice, we will need to distinguish the small dimensional cases ($m = O(1)$) from "large" dimensional cases ($m = O(10^2)$ and much more).

The challenging case will be the large dimensional one of course. That is, a key question will be:

> *How to compute the m values $< \nabla j(u), \delta u >$ for a large number of directions δu ? (m large)*

In the next paragraphs, we first present methods to compute $< \nabla j(u), \delta u >$ which are tractable for m very small only, that is for very small dimensional inverse problems only.

11.5.2 Computing the gradient without adjoint model

Two natural options arise to compute the differential $j'(u)$, therefore the gradient $\nabla j(u)$.

Option 1: the Finite Difference gradient

As already mentioned, the historical method, and the most simple one too, consists to approximate the gradient values using Finite Differences (FD).
Let U_h be the discrete control space, $\dim(U_h) = m$. Then, *given* $\delta u \in \mathbb{R}^m$, an approximation of the gradient in the direction δu can be obtained by employing one of the three following formulas.

$$j'(u) \cdot \delta u \approx \pm \frac{j(u \pm \varepsilon \delta u) - j(u)}{\varepsilon} \text{ at order 1 in } \varepsilon \tag{11.22}$$

$$j'(u) \cdot \delta u \approx \frac{j(u + \varepsilon \delta u) - j(u - \varepsilon \delta u)}{\varepsilon} \text{ at order 2 in } \varepsilon \tag{11.23}$$

Advantages and drawbacks of the FD approach.
$\oplus$: simple to implement, non-intrusive.
$\ominus$: requires $(m+1)$ evaluations of $j(u)$ therefore $(m+1)$ resolutions of the direct model. This is generally not possible for m large.
$\ominus$: The accuracy depends on the choice of ε (and an optimal value of ε is a-priori unknown).

Option 2: expression of $j'(u)$ based on the Tangent Linear Model (TLM)

The straightforward expression of $j'(u)$ (differential calculations) Let us write the straightforward expression of the differential. Let u_0 in U, for all $\delta u \in U$,

$$\boxed{j'(u_0) \cdot \delta u \;=\; \frac{\partial J}{\partial u}(u_0; y(u_0)) \cdot \delta u + \frac{\partial J}{\partial y}(u_0; y(u_0)) \cdot w(u_0; \delta u)} \tag{11.24}$$

where $w(u_0; \delta u)$ denotes the derivative of the state y with respect to u in the direction δu:

$$w(u_0; \delta u) = \frac{dy}{du}(u_0) \cdot \delta u \tag{11.25}$$

In the case that $J(u; y)$ has the particular form (11.3), we have: $\partial_y J(u_0; y) = J'_{obs}(y)$ and $\partial_u J(u_0; y) = \alpha_{reg} J'_{reg}(u_0)$. Therefore:

$$\boxed{j'(u_0) \cdot \delta u \;=\; J'_{obs}(y(u_0)) \cdot w(u_0; \delta u) + \alpha_{reg} J'_{reg}(u_0) \cdot \delta u} \tag{11.26}$$

$w(u; \delta u)$ represents the differential of the state with respect to the control variable.
For example, $w(u; \delta u)$ represents the differential of a temperature field in the domain with respect to the (inhomogeneous therefore spatially distributed) diffusivity parameter.
particular This quantity $w(u; \delta u)$ is not intuitive however it can be obtained by simply deriving the direct model: this is the so-called *Tangent Linear Model* (TLM).

The TLM The TLM consists to differentiate the state equation with respect to the control variable u. By simple differentiation, we obtain:

$$\frac{\partial A}{\partial u}(u; y^u) \cdot \delta u + \frac{\partial A}{\partial y}(u; y^u) \cdot (\frac{dy}{du}(u) \cdot \delta u) = F'(u) \cdot \delta u \tag{11.27}$$

Therefore the TLM:

$$\boxed{\left\{\begin{array}{l} \text{Given } u_0 \in U \text{ and } y^{u_0} \text{ the corresponding solution of the state equation (11.1),} \\ \text{ given a direction } \delta u \in U, \text{ find } w(u_0; \delta u) \in V \text{ such that:} \\ \dfrac{\partial A}{\partial y}(u_0; y^{u_0}).w(u_0; \delta u) = \left[F'(u_0) - \dfrac{\partial A}{\partial u}(u_0; y^{u_0})\right] \cdot \delta u \text{ in } \Omega \\ \text{with corresponding linearized boundary conditions on } \partial\Omega \end{array}\right.} \tag{11.28}$$

Remark 11.11.

- *For each new value of δu, only the RHS of the TLM changes.*
  As a consequence if the numerical solver relies on a factorization of the LHS, the latter can be done once for all.
- *In the case of a linear model, that is the map $y \mapsto A(\cdot; y)$ is linear, the TLM simplifies as:*

$$A(u_0; w(u_0; \delta u)) = \left[F'(u_0) - \frac{\partial A}{\partial u}(u_0; y^{u_0})\right] \cdot \delta u \textit{ in } \Omega$$

In this case, the differential operator therefore the numerical solver, are the same to the direct model solver. Only the RHS changes compared to the direct model.

The *weak form* of the TLM is as follows:

$$\boxed{\left\{\begin{array}{l} \text{Given } u \in U \text{ and } y^{u_0} \text{ solution of (11.9),} \\ \text{ given } \delta u \in U, \text{ find } w \in V \text{ such that:} \\ \dfrac{\partial a}{\partial y}(u; y^{u_0}, z).w = [\dfrac{\partial b}{\partial u}(u; z) - \dfrac{\partial a}{\partial u}(u; y^{u_0}, z)].\delta u \text{ for all } z \in V \end{array}\right.} \tag{11.29}$$

$$\boxed{\begin{array}{l} \text{Solving the TLM provides } w(u; \delta u) = \dfrac{dy}{du}(u).\delta u, \\ \text{that is the derivative of the state } y \text{ with respect to the control } u \text{ in the direction } \delta u. \end{array}}$$

Recall that: $\mathcal{M} : u \in U \mapsto y^{u_0} \in V$. Therefore $\frac{dy}{du}(u) \in \mathcal{L}(U; V)$ and $w(u; \delta u) \in V$.

Note that of course if the direct model is linear then we simply have: $\frac{\partial A}{\partial y}(u_0; y^{u_0}).w = A(u_0; w)$.

Advantages and drawbacks of the TLM-based expression.

$\ominus$: If the direct model is non-linear, the TLM has to be implemented (intrusive approach).
Note that if the non-linear model is solved by the Newton-Raphson method (or if the direct model is linear), then the RHS only has to be coded.
The TLM has to be solved m times to obtain $w(u;\delta u)$ in each direction δu. Therefore if m is large and the CPU time for each resolution is large than the TLM approach to compute $w(u;\delta u)$ is prohibitive.

$\oplus$: The accuracy of $w(u;\delta u)$, therefore of the gradient, is fully controlled by the numerical scheme accuracy.
Compared to the FD approach, this does not depend on an arbitrary setting of ε.

In the end, the FD approach and the TLM approach are feasible for small dimension cases only that is for $m = O(1)$. Moreover, if possible, it is preferable to compute the gradient using the TLM compared to the FD.

Remark 11.12. *It is assumed that the TLM is well-posed. Let us remark that if we have proved existence of solutions to the non-linear model, one likely had to prove that the linearized model is well-posed.*
Moreover, if the implicit function theorem holds (and has been applied to prove the differentiability of the state with respect to the control), then the linearized problem must be well-posed.
Nevertheless for real-like non-linear problems, even the linearized model analysis can be non straightforward at all...

Resulting sensitivity maps Let us point out that the TLM provides $w(u;\delta u)$ that is the *local sensitivity of the state y^u with respect to the control u*, at "point" u in the direction δu.
In a modeling context, the resulting sensitivity map (it is distributed values) constitue rich information to better understand the model and/or the modeled phenomena

The gradient obtained from the TLM correspond to the so-called "sensitivity functions" derived in the books [15, 27].

Exercice 11.13. *Write the TLM of your practical; both the weak and the classical forms.*

Exercice 11.14. *In your programming practical, write a formal procedure aiming at plotting "local sensitivity maps".* □

11.5.3 Gradient components: in the weak or the classical form? *

This a "to go further paragraph".
We have written above the state, linear tangent and adjoint equations in their weak forms for few reasons. First, it is the right framework to study the solutions in the weak sense. Second,

deriving the correct adjoint equations in the weak form may be more clear since the weak forms include naturally the boundary conditions. Third, it is the natural framework in the case of the finite element method.
Nevertheless, in practice if not considering the finite element method, deriving the equations in their weak forms is not compulsory. Once the reader is confortable with these equation derivations process, it is possible to write all the required equations directly in the classical forms or going back from the weak to the classical forms.
A derivation of the equations in finite dimension or in the semi-discrete form (time-dependent problems) are proposed in Section **??**.

Gradient discretization in the case of FEM In the case of Finite Element discretization, an extra question remains concerning the discretization of the gradient expression (**??**). Recall $j'(u) \in \mathcal{L}(U;\mathbb{R})$. Let us denote $\{\psi_i^u\}_{1\leq i\leq m}$ the finite element basis of the control variable u. Then the i-th component of the (discretized) gradient is naturally defined as follows :

$$\partial_i j(u) =< j'(u), \psi_i^u >$$

Example Let us consider the direct model : $-\Delta y = u$ (the control is the RHS) with homogeneous Dirichlet boundary conditions on $\partial\Omega$. Let us consider the cost function :

$$j(u) = \frac{1}{2}\int_\Omega (y^u)^2 dx + \frac{1}{2}\int_\Omega u^2 dx$$

If using finite element discretization, then the ith component of the gradient reads :

$$\partial_i j(u) = \int_\Omega p^u \psi_i^u dx + \int_\Omega u\psi_i^u dx \quad , \ 1 \leq i \leq m$$

with p^u the (discrete) adjoint state function (to be decomposed in its own finite element basis) and u to be decomposed in its finite element basis too.

Finite difference case Now let us consider the same problem but using finite difference schemes (and the same discretization for u and p, with m point values). The ith component of the (discrete) gradient reads :

$$\partial_i j(u) = p_i^u + u_i \ \ , \ 1 \leq i \leq m$$

with p_i^u the ith value of the (discrete) adjoint state.

In the case of *finite element discretization*, there is a choice to make for the gradient definition. In the example above, the choice would read:

$$\boxed{\int_\Omega p^u \psi_i^u dx \text{ vs } p_i^u} \tag{11.30}$$

These two possible expressions do not present the same properties, in particular concerning their dependence on the local mesh element size.

11.6 Cost gradient computation: the adjoint method

Previously, the optimality system has been formally derived by introducing the Lagrangian and by calculating necessary conditions of its stationary points. One of the resulting equation was new; it was the so-called adjoint equation.
In this section, first the adjoint model is derived in a different way, second the calculations are rigorously justified (Theorem 11.15).

The adjoint equations are a mathematical trick enabling the gradient computation by solving one (1) extra system only. This has to be compared to the $O(m)$ resolutions if using the FD-based approach or the TLM-based approach.

11.6.1 Deriving the gradient expression without the term $w(u;\delta u)$

In this section the expression of the adjoint equations and the corresponding gradient expression are rigorously derived for the general direct model (11.1), that is:

$$A(u;y) = F(u) \text{ in } \Omega \text{ with BCs on } \partial\Omega$$

The goal is to minimize $j(u)$ with $j(u) = J(u;y^u)$, y^u the unique solution of the direct model.

Remarks

- Below, we denote indifferently:
  o $j'(u)\cdot\delta u \equiv < j'(u), \delta u >_{U'\times U}$ with $< .,. >_{U'\times U}$ the duality product in U (U Banach space),

  o $G(y)\cdot z \equiv < G(y), z >_{V'\times V}$ with $< .,. >_{V'\times V}$ the duality product in V.

- If not confortable with the employed mathematical notations, the reader may directly read the resulting expressions in discrete form in the summary section 11.7.

Recall that: $\forall \delta u \in U$,

$$j'(u)\cdot\delta u \;=\; \frac{\partial J}{\partial u}(u;y^u)\cdot\delta u + \frac{\partial J}{\partial y}(u;y^u)\cdot w(u;\delta u) \tag{11.31}$$

Recall the TLM:

$$< \frac{\partial A}{\partial y}(u;y^u)\cdot w(u;\delta u), z >_{V'\times V} = < \frac{\partial F}{\partial u}(u).\delta u, z >_{V'\times V} - < \frac{\partial A}{\partial u}(u;y^u)\cdot\delta u, z >_{V'\times V} \quad \forall z \in V \tag{11.32}$$

with $w(u;\delta u)$ defined by (11.25).

Recall the relation characterizing the adjoint operator of a *linear* operator L:

$$\forall (y,z) \in V \times V, \; < Ly, z >_{V' \times V} = < L^* z, y >_{V' \times V}$$

By adding the two equations above and by using the definition of an adjoint operator, we obtain: $\forall \delta u \in U, \; \forall z \in V$,

$$\begin{aligned} j'(u)\cdot \delta u &= \frac{\partial J}{\partial u}(u; y^u)\cdot \delta u - < \left(\frac{\partial A}{\partial u}(u;y^u) - \frac{\partial F}{\partial u}(u) \right)^* \cdot z, \delta u >_{U' \times U} \\ &\quad + < \left(\frac{\partial J}{\partial y}(u;y^u) - \left(\frac{\partial A}{\partial y}(u;y^u) \right)^* \cdot z \right), w(u;\delta u) >_{V' \times V} \end{aligned} \tag{11.33}$$

where $(\square)^*$ denotes adjoint operators, in particular $(\partial_y A)^*$ is the adjoint operator of the *linearized* direct model operator $\partial_y A$.

The goal is now to make vanish the term in $w(u;\delta u)$ in the expression of $j'(u)\cdot \delta u$ above.

To do so, we define the so-called *adjoint field* p^u such that it satisfies:

$$< \left(\frac{\partial A}{\partial y}(u;y^u) \right)^* \cdot p^u, w >_{V' \times V} \;=\; < \frac{\partial J}{\partial y}(u, y^u), w >_{V' \times V} \qquad \forall w \in V \tag{11.34}$$

We then reach our goal: an expression of $j'(u)$ *independent of* $w(u;\delta u)$.
The obtained expression writes: $\forall \delta u \in U$,

$$< j'(u), \delta u >_{U' \times U} \;=\; < \frac{\partial J}{\partial u}(u;y^u), \delta u >_{U' \times U} - < \left(\frac{\partial A}{\partial u}(u;y^u) - \frac{\partial F}{\partial u}(u) \right)^* \cdot p^u, \delta u >_{U' \times U} \tag{11.35}$$

Therefore the explicit expression of $j'(u)$ in $U' = \mathcal{L}(U, \mathbb{R})$,

$$j'(u) \;=\; \frac{\partial J}{\partial u}(u;y^u) \;-\; \left(\frac{\partial A}{\partial u}(u;y^u) - \frac{\partial F}{\partial u}(u) \right)^* \cdot p^u \quad \text{in } U' \tag{11.36}$$

This expression of $j'(u)$ independent of $w(u;\delta u)$ relies on the so-called adjoint equation (11.34).

Note that we have denoted indifferently $j'(u)\cdot \delta u \equiv < j'(u), \delta u >_{U' \times U}$ with $< .,. >_{U' \times U}$ the duality product in U (U Banach space).

11.6.2 The general key result

The formal calculations above have shown the key expressions of the adjoint equation and the resulting gradient (differential) expression (11.36). Below, these same calculations are re-demonstrated by using weak forms too. Moreover, a few comments are made.

Theorem 11.15. *Let us consider the direct model (11.1) and the cost function $j(u)$ defined by (11.3)-(11.4). It is assumed that:*
i) the state equation (11.9) is well-posed,
ii) the TLM (11.29) is well-posed,
iii) the operators $A(u;y)$, $F(u)$, see (11.9), are C^1 with respect to u.
Then, given a C^1 objective function $J(u;y)$ and the cost function $j(u)$ defined by (11.4), the cost function $j(c)$ is of class C^1.
Moreover, the expression of the differential $j'(u)$ satisfies: for all δu in U,

$$j'(u) \cdot \delta u \;=\; \frac{\partial J}{\partial u}(u;y^u) \cdot \delta u \;-\; \left(\frac{\partial a}{\partial u}(u;y^u,p^u) \cdot \delta u - \frac{\partial b}{\partial u}(u,p^u) \cdot \delta u \right) \tag{11.37}$$

with $\partial_u J(u;y^u) = \alpha_{reg} J'_{reg}(u)$ if considering the particular decomposition (11.3) of $J(u;y)$.
y^u is the unique solution of the state equation (11.9),
and p^u is solution of the adjoint equation:

$$\left\{ \begin{array}{l} \textit{Given } u \textit{ and } y^u \textit{ the unique solution of (11.9), find } p \in V \textit{ satisfying:} \\ \dfrac{\partial a}{\partial y}(u;y^u,p) \cdot z = \dfrac{\partial J}{\partial y}(u,y^u) \cdot z \quad \textit{for all } z \in V \end{array} \right. \tag{11.38}$$

with $\partial_y J(u;y^u) = J'_{obs}(y)$ if considering the particular decomposition (11.3) of $J(u;y)$.
The adjoint state p^u, solution of (11.38), exists and is unique.

Proof.
Under assumptions i)-iii), the implicit function theorem applies and the differentiability of the state with respect to u follows: the operator $\mathcal{M} : u \in U \mapsto y^u \in V$ is C^1. Therefore $j(u)$ is of class C^1 too.

The linearized problem is well-posed therefore the operator $\partial_y A(u;y^u)$ is an isomorphism from V into V', and its adjoint operator $(\partial_y A)^*(u;y^u)$ is an isomorphism from V into V' too, see e.g. [12]. As a consequence, the adjoint equation (11.34) is well-posed too. $p^u \in V$ is defined as its unique solution.

The adjoint equation reads: $\left(\frac{\partial A}{\partial y}(u;y^u)\right)^* \cdot p = \frac{\partial J}{\partial y}(u,y^u) \quad$ in V'.
Equivalently,

$$\frac{\partial a}{\partial y}(u;y^u,p) \cdot z = \frac{\partial J}{\partial y}(u,y^u) \cdot z \quad \text{for all } z \in V$$

Moreover, for $\delta u \in U$,

$$j'(u) \cdot \delta u \;=\; \frac{\partial J}{\partial u}(u;y^u) \cdot \delta u \;-\; < \left(\frac{\partial A}{\partial u}(u;y^u) - \frac{\partial F}{\partial u}(u) \right) \cdot \delta u, p^u >_{V' \times V}$$

Therefore the final expression (11.37) of $j'(u)$.
□

Advantages and drawbacks of the adjoint-based expression

$\oplus$: The expression of $j'(u) \cdot \delta u$ does not depend on $w(u;\delta u)$ anymore: the expression of $j'(u)$ is explicit with respect to the direction δu.
Thus, after discretization if solving the direct model plus the adjoint model then *all components of the gradient* follow i.e. the complete gradient vector.
In other words, for m large, the adjoint-based approach enables to obtain the m gradient components by one (1) extra system to solve only.

Let us recall that after discretization (in the finite dimension space U_h), we have:

$$\nabla j(u) \in \mathbb{R}^m, \;\; < \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m$$

$\ominus$: The adjoint model has to be implemented (intrusive approach).
This important drawback may be done by automatic differentiation. This option is more or less complex depending on the direct code complexity and the programming langage.

Remarks

- *By construction, the adjoint model is linear*, whatever if the direct model is linear or not. Recall the adjoint is the adjoint operator of the linearized direct operator.

- Let us point out that excepted for few particular cases (e.g. if the operator A is self-adjoint, $A^* = A$, of course), the adjoint model has in the general case no physical meaning.

- If the direct operator is *self-adjoint*, in other words if $a(u,v)$ is bilinear symmetric, then the adjoint operator equals the direct operator (but the RHS).
  Indeed, in such a case, we have:
  $$\partial_y a(u; y^u, p).z \underbrace{=}_{linear} a(u; z, p) \underbrace{=}_{symmetric} a(u; p, z) \tag{11.39}$$
  Only the source term (RHS) and the boundary conditions differ from the state equation. Then the differential operator, hence the numerical method and numerical solver, are the same.

Case of non-homogeneous Dirichlet BCs

Let us consider the condition: $y = y_d$ on $\Gamma_d \subset \partial\Omega$. Then, the direct model reads:

$$\begin{cases} \text{Find } y \in V_t = V_0 \oplus y_d \text{ such that :} \\ a(u; y, z) = b(u; z) \text{ for all } z \in V_0 \end{cases}$$

where V_t, affine subspace, is the Dirichlet translation of V_0 (V_0 subspace of V Hilbert).
A typical example for a second order linear elliptic equation is : $V = H^1(\Omega)$, $V_0 = \{z \in V, \; z =$

0 on $\Gamma_d\}$ and $V_t = V_0 \oplus y_d = \{z \in V,\ z = y_d \text{ on } \Gamma_d\}$.

Then the question is : What the non-homogeneous Dirichlet boundary conditions becomes when defining the TLM hence the adjoint model ?

The answer is : *the non-homogeneous Dirichlet condition in the direct model becomes the corresponding homogeneous condition in the linear tangent and adjoint models.*

Let us show this statement in the linear case.
The direct model, if linear in y, re-reads as follows:

$$\begin{cases} \text{Find } y_0 \in V_0 \text{ such that :} \\ a(u; y_0, z) = b(u; z) - a(u; \tilde{y}_d, z) = \tilde{b}(u; z) \text{ for all } z \in V_0 \end{cases}$$

with $y_0 = y - \tilde{y}_d$, $\tilde{y}_d$ being a raising from y_d on Γ_d onto the whole domain Ω.

Following the proof of Theorem 11.15, it is easy to notice that the corresponding boundary condition in the TLM is homogeneous, hence the same for the adjoint model.

The optimality system

In the case, $K = U_{ad} = V$ and if considering the particular decomposition (11.3) of $J(u; y)$, the optimality system reads as follows.
The optimal control solution u of Problem (11.7) has to satisfy:

$$\boxed{\begin{cases} a(u; y^u, z) \;=\; b(u; z) \quad \forall z \in V \\ \partial_y a(u; y^u, p) \cdot z \;=\; J'_{obs}(y^u) \cdot z \quad \forall z \in V \\ j'(u) \cdot \delta u \;=\; 0 \quad \forall \delta u \in U \\ \quad \text{with } j'(u) \cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u) \cdot \delta u - (\partial_u a(u; y^u, p^u) - \partial_u b(u; p^u)) \cdot \delta u \end{cases}} \tag{11.40}$$

The optimality system (11.40) is nothing else than the stationary point conditions (11.14) of the Lagrangian.
The adjoint state p is the lagrangian multiplier associated to the "model constraint".

Remark 11.16. ** ("To go further"). The adjoint equations for a coupled system. If the direct model is composed by two PDEs equations weakly coupled then the adjoint system is composed by the corresponding adjoint equations weakly coupled too by in the reverse way.*
If the direct model is composed by two PDEs equations coupled (fully) then the adjoint system is composed by the corresponding adjoint equations (fully) coupled too.

Numerical resolution: the iterative algorithm 3D-Var

If the dimension of the control variable u is large, gradient-based local minimization methods only are affordable (versus the Newton method).

We refer to the discussion in Section 9.4.
In this case, the employed algorithm to approximate the solution u^* is the 3D-Var algorithm, see Fig. 9.1.

Exercises

Exercice 11.17. *Write the adjoint of a few classical second order and first order operators, in the case of:*

- *Dirichlet BC all over the boundary,*
- *Mixed B.C.*

See details of the enunciation in the supplementary material.

Exercice 11.18. *Write the optimality system which characterizes the solution u of the optimal control problem of your practical. Detail both the weak and the classical forms.*

11.7 The fundamental equations at a glance

11.7.1 General continuous formalism

The considered general **non-linear stationary PDE model** reads:

$$\left\{\begin{array}{l} \text{Given } u(x), \text{ find } y(x) \text{ such that:} \\ A(u(x);y(x)) = F(u(x)) \text{ in } \Omega \\ \text{with Boundary Conditions on } \partial\Omega \end{array}\right. \tag{11.41}$$

In weak form:

$$u \in V_t : \; a(u; y^u, z) \;=\; b(u, z) \quad \forall z \in V_0 \tag{11.42}$$

The direct model (= the state equation) is supposed to be well-posed.

The parameter-to-state operator ("model operator") $\mathcal{M}(u)$ is defined as: $\mathcal{M}(u) = y^u$.
This operator $\mathcal{M}(\cdot)$, which is a-priori non-linear, is supposed to continuous (well-posed direct model).

The cost function $j(u)$ is defined from the observation function $J(u;y)$ as follows:

$$j(u) = J(u\,;y^u) \tag{11.43}$$

where $y^u(x)$ denotes the unique solution of the direct model, given $u(x)$.

The observation function $J(u;y)$ is classically decomposed as follows:

$$J(u\,;y) \;=\; J_{obs}(y) \;+\alpha_{reg}\; J_{reg}(u) \tag{11.44}$$

with the data misfit term:

$$J_{obs}(y) \;=\; \frac{1}{2}\Big\| \,Z(y) - z_d \Big\|^2_{R^{-1}} \tag{11.45}$$

The observation operator $Z(\cdot)$ may be linear or not.
The regularization term is here defined from quadratic terms (in u or higher-order terms).
Classical expressions are: $J_{reg}(u) \;=\; \frac{1}{2}\Big\| \,u - u_b \Big\|^2_{B^{-1}}$ or $\frac{1}{2}\Big\| \,D^p u \Big\|^2_0$ with $p = 1$ or 2.

The inverse problem is formulated as the following **optimization problem**:

$$\left\{\begin{array}{l} \text{Minimize } j(u) \text{ in } U_{ad} \text{ under the "model constraint"} \\ \text{ since } j(u) = J(u;y^u) \text{ with } \; y^u = \mathcal{M}(u). \end{array}\right. \tag{11.46}$$

The adjoint model reads:

$$\left\{\begin{array}{l} \text{Given } u(x), \text{given } y^u(x), \text{ find } p(x) \text{ such that:} \\ (\partial_y A(u;y^u)(x))^* \cdot p(x) \;=\; J'_{obs}(y^u(x)) \text{ in } \Omega \\ \text{with the } \underline{\text{adjoint B.C.}} \text{ on } \partial\Omega \end{array}\right. \tag{11.47}$$

In weak form. By defining $a^*((u,y^u);p,z) \equiv \partial_y a(u;y^u,p)\cdot z$, the adjoint equation reads

$$p \in V_0 : \; a^*((u,y^u);p,z) \;=\; J'_{obs}(y^u)\cdot z \quad \forall z \in V_0 \tag{11.48}$$

The resulting gradient expression.
The relationship between the gradient $\nabla j(u)$ and the differential $j'(u)$ is:

$$< \nabla j(u), \delta u >_{\mathbb{R}^m} = j'(u) \cdot \delta u \text{ for all } \delta u \in U_h \subset \mathbb{R}^m \tag{11.49}$$

The differential of j reads: for all $\delta u \in U$,

$$j'(u) \cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u) \cdot \delta u - (\partial_u A(u; y^u) \cdot \delta u - F'(u) \cdot \delta u) \cdot p^u \tag{11.50}$$

In weak form.

$$j'(u) \cdot \delta u \;=\; \alpha_{reg} J'_{reg}(u) \cdot \delta u - \partial_u a(u; y^u, p^u) \cdot \delta u + \partial_u b(u; p^u) \cdot \delta u \tag{11.51}$$

In the case of a composite gradient i.e. c containing different components, $u = (u_1, u_2)$, we have:

$$j'(u) \equiv (\frac{\partial j}{\partial u_1}(u), \frac{\partial j}{\partial u_2}(u))^T \text{ and } j'(u) \cdot \delta u = \frac{\partial j}{\partial u_1}(u) \cdot \delta u_1 + \frac{\partial j}{\partial u_2}(u) \cdot \delta u_2$$

11.7.2 Discrete formalism

Recall that the general continuous formalism above, based on the weak form of the equations, enables to rigorously derive the adjoint equations and the gradient expression, including in the non-linear case, even if the boundary conditions are non trivial. The discrete formalism below is easier to handle but it may be incomplete in presence of complex boundary conditions.

The direct model reads:

$$\left\{ \begin{array}{l} \text{Given } u \in \mathbb{R}^m, \text{ find } y \in \mathbb{R}^n \text{ such that:} \\ A(u; y) = F(u) \end{array} \right. \tag{11.52}$$

$A(u; y)$ represents a system of n (non-linear) equations at n unknowns $(y_1, \ldots, y_n)$.
One has: $A : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$; $A(u; y) \in \mathbb{R}^n$. $F : \mathbb{R}^m \to \mathbb{R}^n$; $F(u) \in \mathbb{R}^n$.

The solution of this non-linear system is supposed to be unique and is denoted by y^u.
Let us denote by $D_y A(u; y^u)$ (resp. $D_u A(u; y^u)$) the n equations derived with respect to y (resp. u): it is the $n \times n$-Jacobian matrix (resp. the $n \times m$-Jacobian matrix).
Similarly, we denote by $D_u F(u)$ the n components of $F(u)$ derived with respect to u: it is the $n \times m$-Jacobian matrix.

Let us recall the adjoint property (transpose in the real case) of a $n \times n$-matrix M (therefore a linear operator from $\mathbb{R}^n$ onto $\mathbb{R}^n$): for y and z in $\mathbb{R}^n$, $(M^T y, z)_{\mathbb{R}^n} = (y, Mz)_{\mathbb{R}^n}$.
The observation function $J(u; y)$ is defined as indicated in the general case.

Given these conventions, one has the adjoint model which reads:

$$\left\{\begin{array}{l} \text{Given } u \in \mathbb{R}^m, \text{given } y^u \in \mathbb{R}^n, \text{ find } p \in \mathbb{R}^n \text{ such that:} \\ (D_y A(u; y^u))^T p \;=\; J'_{obs}(y^u) \\ \text{(modulo the Dirichlet boundary conditions)} \end{array}\right. \tag{11.53}$$

This is a linear system of n equations at n unknowns $(p_1, \dots, p_n)$. The solution of this linear system is supposed to be unique and is denoted by p^u.

The resulting gradient $\nabla j(u) \in \mathbb{R}^m$ reads:

$$\nabla j(u) \;=\; \alpha_{reg} \nabla J_{reg}(u) - (D_u A(u; y^u))^T\, p^u + (D_u F(u))^T\, p^u \tag{11.54}$$

In the case of few components e.g. $u = (u_1, u_2)\mathbb{R}^{m_1} \times \mathbb{R}^{m_2}$, then one simply has:

$$\nabla_u j(u) = (\nabla_{u_1} j(u), \nabla_{u_2} j(u))^T$$

11.8 Applications to classical PDEs and operators

Optimality system for different BVP / examples

11.8.1 Classical PDEs

Diffusion equations

Linear case

Non-linear diffusivity

Advection-diffusion equations

Linear case

Viscous Burger's model

Elasticity system

11.8.2 Adjoint of classical operators

ToDo: Ecrire catalogue de termes !

Chapter 12

VDA for Time-Dependent PDEs

This chapter aims at extending the VDA formulation to unsteady PDEs. The latter can be (well-posed) parabolic or hyperbolic equations, non-linear or not. The calculations derived in all this chapter are formal in the sense that we do not pay attention to the functional spaces. For more rigorous derivations of optimality systems in optimal control problems, the reader may consult e.g. [32, 41].

As in the stationary case, the derivations are first presented in a discrete formalism since it does not require demonstrated skills in differential calculus. However, these derivations in finite dimensions are restrictive. Next, the adjoint equation and the gradient expression are derived in a general case therefore enabling to use the formula for a particular problem, see Theorem 12.4.

The resulting algorithm obtained for a time-dependent system is classically called 4D-var (even if the model is not 3D in space...). Some strategies to reduce the complexity of this CPU and memory consuming algorithm is discussed. Finally real-world applications are presented

The outline of this chapter is as follows[1].

Contents

[1] Recall that the sections indicated with a * are "to go further sections". They can be skipped in a first reading or if the reader is not particularly interested in deeper mathematical basis, mathematical proofs.

12.1 The inverse formulation

12.1.1 The general direct model

Let us consider a general unsteady PDE model however first order in time to simplify the presentation:

$$\boxed{(\mathcal{D}) \begin{cases} \text{Given the I.C. } y_0(x)\text{, given the space-time control } u(x,t), \\ \text{ find the state } y(x,t) \text{ satisfying:} \\ \partial_t\, y(x,t) \;+\; A(u(x,t)\,; y(x,t)) \;=\; F(u(x,t)) \qquad \text{in } \Omega\times]0,T[\\ y(x,0) \;=\; y_0(x) \text{ in } \Omega \\ \text{with Boundary Conditions for all } t \end{cases}} \tag{12.1}$$

where $A(u; y)(x, t)$ is the differential operator. $F(u(x, t))$ is the RHS which may depend on the control too.

Examples of direct models As a scalar parabolic equation example, the reader may guess to the heat equation (scalar linear parabolic equation) or to the non-linear case:

$$A(u; y) = -div(\lambda(u_1; y)\nabla y) + w \cdot \nabla y + cy \text{ and } F(u) = u_2$$

with $u = (u_1, u_2)$.

One may guess to the (viscous) Navier-Stokes equations (non-linear parabolic system) too or EXAMPLE STRUCTURAL (non-linear parabolic system) or to the Saint-Venant equations (non-linear hyperbolic system).

In real-world problems, the I.C. is often uncertain. The I.C. can even be the most important "parameter" to be identified/estimated e.g. in atmosphere dynamic problems for weather prediction. Then, we consider the control variable enriched with the I.C. as:

$$\boxed{c(x,t) = (y_0(x), u(x,t))} \tag{12.2}$$

Of course, the solution $y(x, t)$ of $(\mathcal{D})$ depends on the I.C. $y_0(x)$ and on the parameter $u(x, t)$.

In all the sequel, the state is denoted as $y(c; t)$, $y(t)$, $y(c)$ or simply y, depending on the context.

We assume that the direct model is well posed in the following sense.

Assumption 12.1. *Given $c(x, t) \in \mathcal{C}$ and $T > 0$, it exists a unique function $y(x, t)$, $y \in W_V(0, T)$, solution of Problem $(\mathcal{D})$. Furthermore, this unique solution y depends continuously on $c(x, t)$.*

Mathematical functional spaces* Typical functional spaces $\mathcal{C}$ and $W_V(0,T)$ are as follows. Let denote by V be a Hilbert space e.g. the Sobolev space $H_0^1(\Omega)$ for a scalar second order linear PDE as the heat equation (linear parabolic equation).
The time-dependent PDE is assumed to be first order in time then the considered state space W (a space-time functional space) is classically defined as:

$$W_V(0,T) = \{f \ s.t. \ f \in L^2(0,T;V), \ \partial_t f \in L^2(0,T;V')\}$$

The control parameter space $\mathcal{U}$ is supposed to be a Hilbert space. Typically, one considers: $\mathcal{U}_T = L^2(0,T;\mathcal{U})$ with $\mathcal{U}$ a Banach space e.g. $L^\infty(\Omega)$. The norm of $\mathcal{U}_T$ is defined from the scalar product of $\mathcal{U}$ as:
$\langle u_1, u_2 \rangle_{\mathcal{U}_T} \ = \ \int_0^T \langle u_1(t), u_2(t) \rangle_{\mathcal{U}} dt.$
Let H be the Hilbert space where the I.C. lives in. Then, the control $c(x,t)$ belongs to the space $\mathcal{C} = H \times \mathcal{U}$.

The reader may refer e.g. to [33]. The examples of functional spaces above are typical ones for linear BVP. Moreover, the control space $\mathcal{U}_T$ may be relaxed (i.e. imposing less regularity).

Assumptions of differentiability* The parameter-to-state operator reads here as:

$$\boxed{\mathcal{M} : \mathcal{C} \rightarrow \ W_V(0,T) : c(x,t) = (y_0(x), u(x,t)) \mapsto y(c(x,t);x,t)} \tag{12.3}$$

The direct model is well-posed implies that $\mathcal{M}(y)$ is continuous, for all t.

To consider cost functions $j(c)$ of class C^1 (continuously differentiable), one needs to assume that the state y is differentiable with respect to the control parameter c that is

Assumption 12.2. *The model operator $\mathcal{M}(c)$ is continuously differentiable for all $t \in \,]0,T[$.*

Under the assumption above, one can formally write:

$$y(c + \delta c; t) \ = \ y(c;t) + \frac{dy}{dc}(c;t) \cdot \delta c + o_0(\|\delta c\|_c) \tag{12.4}$$

This differentiability property will be necessary to calculate the cost function differential $j'(c)$. In some cases, this differentiability property is not satisfied at all control value c.

Indeed, in the case of a non-linear hyperbolic system such as the Euler (or Shallow Water) equations for example, a shock can appear when making varies continuously e.g. a physical model parameter or a boundary condition value. In this case, the operator $\mathcal{M}(c)$ is even not continuous, therefore cannot be differentiable.

12.1.2 Cost function terms: data misfit and regularizations

The objective function J is decomposed as in the stationary case:

$$\boxed{J(c\,;y) \;=\; J_{obs}(y) \;+\; \alpha_{reg} J_{reg}(c)} \tag{12.5}$$

Next, the cost function j to be minimized is defined from J as usual:

$$\boxed{j(c(x,t)) = J(c(x,t)\,;y^c(x,t)))} \tag{12.6}$$

where $y^c(x,t)$ is the (unique) solution of the direct model $(\mathcal{D})$, given $c(x,t)$.
We have: $j : \mathcal{C} \to \mathbb{R}$.

Data misfit term In the unsteady case, the misfit term J_{obs} is naturally integrated in time as follows:

$$J_{obs}(y(x,t)) \;=\; \int_0^T \left\| \, Z(y(x,t)) - z^{obs}(x,t) \right\|^2_{R^{-1}} \, dt \tag{12.7}$$

with $Z(\cdot)$ the observation operator.
Z is a-priori non linear. R^{-1} denotes a norm like in the stationary case. In practice, as already mentioned, it is often a diagonal matrix because of lack of information.

In practice, observations are often local ("point-wise"), moreover very sparse. Then, in the case the observations are provided as time-series, the misfit observation term $J_{obs}(y)$ reads as:

$$\boxed{J_{obs}(y) \;=\; \sum_{n=1}^{N} \left\| \, Z(y(t_n^{obs})) - z_n^{obs} \right\|^2_{R^{-1}}} \tag{12.8}$$

where z_n^{obs} is the measurement at instant t_n^{obs}, $n = 1, \ldots, N$.

The complete dataset is then: $z^{obs} = \{z_n^{obs}\}_{n=1,\ldots,N}$.

Regularization terms Recall that $c(x,t) = (y_0(x), u(x,t))$. Then, we naturally consider a regularization term for each control component as:

$$\boxed{J_{reg}(c(x,t)) = J^0_{reg}(y_0(x)) + J^u_{reg}(u(x,t))} \tag{12.9}$$

If considering a background value y_b for the I.C. to attract the minimization algorithm to this value, we set: $J^0_{reg}(y_0) = \left\| \, y_0 - y_b \right\|^2_{B^{-1}}$.

The norm B^{-1} may be defined to represent "at best" the inverse of the covariance matrix of the background error matrix B as already discussed, see Section **??**.

This term is quadratic in y_0 therefore strictly convex.

For the control parameter $u(x,t)$, one considers the same regularization terms as in the stationary case (see Section 11.1.3).
We may define $J^u_{reg}(u)$ from a background value u_b: $J^u_{reg}(u) = \left\| u - u_b \right\|_2^2$. This choice comes with its consequences: this defines a strictly convex term in u and it attracts the solution to the background value, however for the best or for the worst...

One can consider higher-order terms too as in the stationary case e.g. $J^u_{reg}(u) = \left\| \nabla u \right\|_2^2$. In this case, $J^u_{reg}(u)$ is no longer convex with respect to u but with respect to ∇u only.

12.1.3 The optimization problem

The optimization problem writes similarly to the stationary case:

$$\boxed{\begin{cases} \text{Find } c^*(x,t) = (y_0^*(x), u^*(x,t)) \in H \times \mathcal{U}_T \text{ such that:} \\ j(c^*) = \min\limits_{H\times\mathcal{U}_T} j(c) \end{cases}} \qquad (12.10)$$

The control parameter vector c is here a-priori time-dependent. As a consequence, its discrete version is an extremely large vector! Therefore the optimization problem is large dimensional.

One of the consequence is that $\nabla j(c)$ has to be computed by employing the adjoint method.

12.2 Optimality equations in finite dimensions (discrete forms)

The derivation of the adjoint equation and the gradient expression in finite dimension is easier than in the continuous form since calculations requires less experience in differential calculus. However, it is harder (and less elegant) to derive the equations for general cases in particular for complex numerical schemes or non classical boundary conditions.
For a sake of simplicity, we present here the optimality equations in the case of a basic explicit one step time scheme (namely the forward Euler scheme). The formal direct model reads:

$$\boxed{(D_d)\begin{cases} \text{Find } \{y_k\}_{1\leq k\leq N_T} \in \mathbb{R}^n \text{ such that :} \\ y_{k+1} = \mathcal{M}_k(y_k) \quad k = 0,1,...,N_T \text{ (the time steps)} \\ y_0 \in \mathbb{R}^n \text{ given.} \end{cases}} \qquad (12.11)$$

where $\mathcal{M}_k(y_k)$ represents the n *non-linear* equations at instant t_k.
We denote by $M_k = D_y\mathcal{M}_k(y_k)$ the linearized model equations at "point" y_k, $M_k \in \mathbb{R}^{n\times n}$.

For a sake of simplicity again, the control variable c is simply here the I.C.:

$$\boxed{c = y_0 \in \mathbb{R}^n}$$

Moreover, it is supposed that: a) the observations $z^{obs} \in \mathbb{R}^m$ are available at all time step; b) the regularization term relies on the knowledge of a good background value y_b.

Then, the cost function reads:

$$j(y_0) = \frac{1}{2}\sum_{k=1}^{N_T}(\mathcal{Z}(y_k) - z_k^{obs})^T R^{-1}(\mathcal{Z}(y_k) - z_k^{obs}) + \alpha_0 \frac{1}{2}(y_0 - y_b)^T B(y_0 - y_b) \tag{12.12}$$

where $\mathcal{Z} : \mathbb{R}^n \to \mathbb{R}^m$ is the observation operator which is a-priori non-linear too.
The norm R is given, $R \in \mathbb{R}^{m\times m}$ (often simply diagonal as already discussed).

We denote the linear tangent observation operator taken at y_k as: $Z_k = D_y\mathcal{Z}(y_k)$, $Z_k \in \mathbb{R}^{m\times n}$.

The cost gradient satisfies: for all δy_0 in $\mathbb{R}^n$,

$$< \nabla j(y_0), \delta y_0 >_{\mathbb{R}^n} = \sum_{k=1}^{N_T} < \Delta_k, Z_k \cdot w_k^\delta >_{\mathbb{R}^m} + \alpha_0 < B(y_0 - y_b), \delta y_0 >_{\mathbb{R}^n}$$

with $w_k^\delta = D_{y_0}(y_k) \cdot \delta y_0$, $w_k^\delta \in \mathbb{R}^n$ and the notation $\Delta_k = R^{-1}(\mathcal{Z}(y_k) - z_k^{obs})$.

By differentiating the direct model, we get:

$$D_{y_0}(y_{k+1}) \cdot \delta y_0 = M_k \cdot D_{y_0}(y_k) \cdot \delta y_0 \ , \ k = 1, ..., N_T \tag{12.13}$$

with the I.C. δy_0 given.
The TLM above reads:

$$w_{k+1}^\delta = M_k w_k^\delta = (M_k \dots M_0)\delta y_0 \tag{12.14}$$

By injecting this into the gradient expression, we obtain:

$$< \nabla j(y_0), \delta y_0 >_{\mathbb{R}^n} = \sum_{k=1}^{N_T} < (M_0^T \dots M_{k-1}^T) Z_k^T \Delta_k, \delta y_0 >_{\mathbb{R}^n} + \alpha_0 < B(y_0 - y_b), \delta y_0 >_{\mathbb{R}^n} \tag{12.15}$$

The 1st term $\sum_{k=1}^{N_T}\left(M_0^T \dots M_{k-1}^T\right) Z_k^T \Delta_k$ in the expression above reads:

$$\begin{aligned} & M_0^T Z_1^T \Delta_1 + (M_0^T M_1^T) Z_2^T \Delta_2 + \dots + (M_0^T M_1^T \dots M_{N_T-1}^T) Z_{N_T}^T \Delta_{N_T} \\ & \qquad = M_0^T \left[Z_1^T \Delta_1 + M_1^T \left[Z_2^T \Delta_2 + [\dots] \, M_{N_T-1}^T Z_{N_T}^T \Delta_{N_T} \right] \dots \right] \qquad (12.16) \\ & \qquad \qquad (12.17) \end{aligned}$$

Let us define the sequence p_k defined as follows: for $k = (N_T - 1), \dots, 0$,

$$p_k = M_k^T p_{k+1} + Z_k^T \Delta_k \tag{12.18}$$

with $p_{N_T} = 0$, and (recall) $\Delta_k = R^{-1}(\mathcal{Z}(y_k) - z_k)$.
We have:

$$\begin{array}{rcl}
p_{N_T-1} & = & Z_{N_T-1}^T \Delta_{N_T-1} \\
p_{N_T-2} & = & M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \\
p_{N_T-3} & = & M_{N_T-3}^T \left[M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \right] + Z_{N_T-3}^T \Delta_{N_T-3} \\
 & \vdots & \\
p_0 & = & M_0^T \left[M_1^T \left[\dots \left[M_{N_T-3}^T \left[M_{N_T-2}^T \left[Z_{N_T-1}^T \Delta_{N_T-1} \right] + Z_{N_T-2}^T \Delta_{N_T-2} \right] + \dots \right] + \dots \right] + Z_1^T \Delta_1 \right] + Z_0^T \Delta_0 \quad (12.19)
\end{array} \tag{12.20}$$

This is a Horner type factorization.
It can be shown that the two expressions (12.16) and (12.19) are equal.
ToDo: calcul pas clair... a verifier.... mais le resultat est bien celui-ci (cf demo continue)
Therefore:

$$p_0 = \sum_{k=1}^{N_T} \left(M_0^T \dots M_{k-1}^T \right) Z_k^T \Delta_k \tag{12.21}$$

In conclusion, by introducing the following sequence (defining the discrete adjoint model):

$$\boxed{(A_d) \left\{ \begin{array}{l} \text{Given the state at each time step } \{y_k\}_{1 \le k \le N_T} \in \mathbb{R}^n, \text{ find } \{p_k\}_{(N_T-1) \ge k \ge 0} \in \mathbb{R}^n \text{ such that :} \\ p_k = M_k^T p_{k+1} + Z_k^T R^{-1}(\mathcal{Z}(y_k) - z_k) \quad \text{for } k = N_T, ..., 1 \text{ (the time steps)} \\ p_{N_T} = 0 \end{array} \right.} \tag{12.22}$$

with $Z_k = D_y \mathcal{Z}(y_k)$, $Z_k \in \mathbb{R}^{m \times n}$, the gradient simply reads as, see (12.15):

$$\boxed{\nabla j(y_0) = p_0 + \alpha_0 B(y_0 - y_b)} \tag{12.23}$$

A few remarks

- The adjoint model is retroacting in time. Its initial condition must be given at final time T.
- The gradient with respect to the Initial Condition equals the adjoint state at initial time (modulo the minus sign and the regularization term).

12.3 The optimality equations in infinite dimension (continuous forms)

In this section, a rigorous derivation of the adjoint equation and the gradient expression are proposed. The general expressions presented in Theorem 12.4, can be applied to a large class of problems.

12.3.1 The TLM-based gradient

The approach is the same as in the stationary case: we derive the cost function $j(c)$, we obtain the differential expression $j'(c)$ in function of the state derivative $w^{\delta c} = \frac{dy}{dc}(c) \cdot \delta c$. The $w^{\delta c}$ is by construction the solution of the Tangent Linear Model (TLM).

Gradient expression depending on the term $w^{\delta c}$

By deriving the cost function (11.4), we get:

$$j'(c) \cdot \delta c \ = \ \partial_c J(c\,; y^c).\delta c + \partial_y J(c\,; y^c).w^{\delta c} \tag{12.24}$$

with $w^{\delta c} = \frac{dy}{dc}(c) \cdot \delta c$ and $c = (y_0, u)$.
If considering the particular form (12.5) (with $\alpha_{reg} = 1$) then:

$$\boxed{j'(c) \cdot \delta c \ = \ J'_{obs}(y^c) \cdot w^{\delta c} + J'_{reg}(c) \cdot \delta c} \tag{12.25}$$

Let us consider the observation term as previously (see Section 12.1.2) with a linear observation operator Z for a sake of simplicity:

$$J_{obs}(y) \ = \ \frac{1}{2} \int_0^T \left\| Zy(t) - z^{obs}(t) \right\|^2_{R^{-1}} dt \tag{12.26}$$

For the regularization term, one can naturally consider:

$$J_{reg}(c) = \frac{1}{2} \alpha_{reg,0} \left\| y_0 - y_b \right\|^2_{B^{-1}} + \frac{1}{2} \alpha_{reg,u} \left\| \nabla u \right\|^2_2 \tag{12.27}$$

with the weights $\alpha_{reg,\square}$ to be determined.
With the definitions above, it follows the expression:

$$j'(c) \cdot \delta c \ = \ \int_0^T \Big\langle \ Z^T R(Z\, y^c(t) - z^{obs}(t)),\ w^{\delta c}(t) \ \Big\rangle \ dt \ + \ \big\langle B^{-1}(y_0 - y_b), \delta y_0 \big\rangle \ + \ \big\langle \nabla u, \nabla(\delta u) \big\rangle \tag{12.28}$$

with $< \cdot, \cdot >$ the corresponding scalar products.

Functional spaces clarification* More rigorously, we have:

$$j'(c) \cdot \delta c \ = \ \int_0^T \Big\langle \ Z^T R \Lambda_{\mathcal{O}}(Z\, y^c(t) - z^{obs}(t)),\ w^{\delta c}(t) \ \Big\rangle_{V' \times V} dt + \big\langle B^{-1}(y_0 - y_b), \delta y_0 \big\rangle_H + \big\langle \nabla u, \nabla(\delta u) \big\rangle_H$$

where $\langle \cdot, \cdot \rangle_{V' \times V}$ denotes the duality product and $\langle \cdot, \cdot \rangle_H$ denotes the H-scalar product.
The operator $\Lambda_{\mathcal{O}}$ is simply the canonical isomorphism from $\mathcal{Z}_0$ into $\mathcal{Z}'_0$ (in finite dimension, it is simply equal to the Identity). Also, $Z^T \equiv Z^* \in \mathcal{L}(\mathcal{Z}'_0, V')$ denotes the adjoint operator of the linear operator Z; it is defined by:

$$\forall \eta \in \mathcal{Z}'_0,\ \forall \xi \in V,\ \big\langle \ Z^* \eta,\ \xi \ \big\rangle_{V' \times V} \ = \ \big\langle \ \eta,\ Z\xi \ \big\rangle_{\mathcal{Z}'_0 \times \mathcal{Z}_0}$$

On the term $w^{\delta c}$ Recall that the (unique) state of the system $y(c)$ is assumed to be differentiable with respect to c, $c = (y_0, u)$.
Given a perturbation $\delta c \in \mathcal{C}$, the term $w^{\delta c}(t)$ represents the state derivative in the direction $\delta c = (\delta y_0, \delta u)$ (Gateaux's derivative). This term satisfies the relation:

$$w^{\delta c} \equiv \frac{dy}{dc}(c) \cdot \delta c \;=\; \frac{\partial y}{\partial y_0}(c) \cdot \delta y_0 \;+\; \frac{\partial y}{\partial u}(c) \cdot \delta u$$

On the differential $j'(c)$ and the composite gradient $\nabla j(c)$ Rigorously speaking, the terminology "gradient" refers to vectors in finite dimension spaces. However, we improperly use the word gradient (as it is usually done in the literature too) to name the differentiable $j'(u) \in \mathcal{L}(H \times \mathcal{U}_T; \mathbb{R})$ or the actual gradient $\nabla j(u) \in \mathbb{R}^m$.

Since the control variables has here two distinct components, $c = (y_0, u)$, then the gradient of $j(c)$ reads:

$$\boxed{\nabla j(c) = (\nabla_{y_0} j(c), \nabla_u j(c))^T} \tag{12.29}$$

And we have the differentiable which satisfies:

$$\boxed{j'(c) \cdot \delta c = \frac{\partial j}{\partial y_0}(c) \cdot \delta y_0 + \frac{\partial j}{\partial u}(c) \cdot \delta u} \tag{12.30}$$

for any perturbation $\delta c = (\delta y_0, \delta u)$.

Recall that we have: $< \nabla j(c), \delta c >_{\mathbb{R}^m} = j'(c) \cdot \delta c$ for all $\delta c \in \mathcal{C}_h \subset \mathbb{R}^m$.

The Tangent Linear Model (TLM)

By deriving the direct model $(\mathcal{D})$ with respect to c (in a direction δc and at "point" $y(c)$), we obtain the TLM.
The TLM solution is the term $w^{\delta c}$. We have:

$$\boxed{(\mathcal{LT}) \begin{cases} \text{Given } c = (y_0, u) \in H \times \mathcal{U}_T \text{ and } y^c \in W_V(0,T) \text{ solution of the direct problem } (\mathcal{D}), \\ \text{given } \delta c = (\delta y_0, \delta u) \in H \times \mathcal{U}_T, \text{ find } w^{\delta c} \in W_V(0,T) \text{ such that:} \\ \partial_t w^{\delta c}(t) \;+\; \frac{\partial A}{\partial y}(u; y) \cdot w^{\delta c}(t) \;=\; -\frac{\partial A}{\partial u}(u; y) \cdot \delta u(t) + F'(u) \cdot \delta u(t) \;\; \forall t \in \,]0, T[\\ w^{\delta c}(0) \;=\; \delta y_0 \end{cases}} \tag{12.31}$$

Remark 12.3. *If the operator A is linear with respect to the state variable y, then we have:* $\frac{\partial A}{\partial y}(u(t), y(t)) \cdot w^{\delta c}(t) = A(u(t), w^{\delta c}(t))$.
In this case, the TLM is the same as the direct model but the I.C. and the RHS.

As a consequence, to compute $w^{\delta c}$ and to obtain the gradient defined by (12.28), one can solve the TLM $(\mathcal{LT})$.

However, the TLM must be solved again to obtain $w^{\delta c}$ for a different perturbation δc.
After discretization, if δc_h is large dimensional, the TLM-based approach is not tractable...
Like in the steady-state case, this is the reason why the adjoint equations are introduced as soon as the discrete control variable dimension is greater than $\mathcal{O}(1)$. Indeed, the adjoint equation will enable to obtain the gradient independently of the dimension of the discrete control variable.

12.3.2 The adjoint-based gradient

We state here the central result providing the expression of the gradient in the general case presented in Section 12.1.1. This gradient expression is based on the adjoint model solution. The result below is the extension of Theorem 11.15 to time-dependent cases.

Theorem 12.4. *Let us consider the direct model (12.1) and the cost function $j(u)$ defined by (12.6)-(12.27).*
It is assumed that:
i) the direct model (12.1) is well-posed,
ii) the TLM (12.31) is well-posed,
iii) the unique solution $y(c)$ is C^1 with respect to c.

Then, the cost function $j(c)$ is C^1 and its gradient components reads $\nabla j(c) = (\partial_{y_0} j(c), \partial_u j(c))^T$ with:

$$\left\{ \begin{array}{rcl} \partial_{y_0} j(c) & = & p^c(0) \; + \; \alpha_{reg,0} (J^0_{reg})'(y_0) \\ \partial_u j(c) & = & - \left[\dfrac{\partial A}{\partial u}(u; y^c) - F'(u) \right]^T p^c(t) + \alpha_{reg,u} (J^u_{reg})'(u) \end{array} \right. \tag{12.32}$$

with y^c the unique solution of the state equation (12.1) and p^c solution of the adjoint model:

$$(\mathcal{A}) \left\{ \begin{array}{l} \text{Given } c(x,t) = (y_0(x), u(x,t)) \in \mathcal{C} \text{ and } y^c(x,y) \in W_V(0,T) \text{ be the unique solution of } (\mathcal{D}), \\ \text{find } p(x,t) \in W_V(0,T) \text{ such that:} \\ -\partial_t \, p(x,t) + \left[\dfrac{\partial A}{\partial y}(u; y^c) \right]^T p(x,t) = J'_{obs}(y^c(x,t)) \;\; \forall t \in \,]0,T[\\ p(x,T) \;\; = \;\; 0 \text{ in } \Omega \end{array} \right. \tag{12.33}$$

The solution p^c of $(\mathcal{A})$ exists and is unique; it is the adjoint state.

Proof.
Under Assumption iii) (deriving from the implicit function theorem, see details in the stationary case), the cost function $j(u)$ is C^1.
First let us recall the direct expression of $j'(c)$, see (12.6)(12.5):

$$j'(c) \cdot \delta c \; = \; J'_{obs}(y^c) \cdot w^{\delta c} \; + \; J'_{reg}(c) \cdot \delta c \tag{12.34}$$

The goal is to obtain an expression of $j'(c)$ independent of $w^{\delta c}$.

The proof follows the same principles than in the stationary case (see the proof of Theorem 11.15). The only difference consists to integrate in time the equations. We write:

$$\int_0^T < (\mathcal{LT}), p >_{V'\times V} dt \;=\; 0 \quad \forall p \in V$$

By integrating by part in time, we get:

$$\begin{aligned}&-\int_0^T \big\langle w^{\delta c}(t), \partial_t p(t)\big\rangle_{V'\times V}\, dt \;+\; \int_0^T \big\langle \tfrac{\partial A}{\partial y}(u;y)\cdot w^{\delta c}(t), p(t)\big\rangle_{V'\times V}\, dt\\ &+ \big\langle p(T), w^{\delta c}(T)\big\rangle_H \;-\; \big\langle p(0), \delta y_0\big\rangle_H \;+\; \int_0^T \big\langle [\tfrac{\partial A}{\partial u}(u;y) - F'(u)]\cdot \delta u(t), p(t)\big\rangle_{V'\times V}\, dt \;=\; 0\end{aligned}$$

By making appear the adjoint operator of $\frac{\partial A}{\partial y}(u;y)$, it comes:

$$\begin{aligned}&\int_0^T \big\langle -\partial_t p(t) + \left[\frac{\partial A}{\partial y}(u;y)\right]^T p(t), w^{\delta c}(t)\big\rangle_{V'\times V} dt\\ &= -\big\langle p(T), w^{\delta c}(T)\big\rangle_H \;+\; \big\langle p(0), \delta y_0\big\rangle_H \;-\; \int_0^T \big\langle [\tfrac{\partial A}{\partial u}(u;y) - F'(u)]\cdot \delta u, p(t)\big\rangle_{V'\times V}\, dt\end{aligned} \tag{12.35}$$

The goal is to make vanish the term $w^{\delta c}(t)$ in (12.34). This goal is reached by setting p as the solution of the following equation:

$$-\partial_t p(t) + \left[\frac{\partial A}{\partial y}(u;y^c)\right]^T p(t) = J'_{obs}(y^c) \quad \forall t \in \,]0,T[\tag{12.36}$$

accompanied with the Initial Condition at final time: $p(T) = 0$.
These equations constitute the adjoint model $(\mathcal{A})$.

The linearized problem is supposed to be well-posed therefore the operator $\partial_y A(u;y^u)$ is an isomorphism from V into V', and its adjoint operator $(\partial_y A)^T(u;y^u)$ is an isomorphism from V into V' too, see e.g. [12]. As a consequence, the adjoint equation above is well-posed too.

By making appear the adjoint operator of $[\frac{\partial A}{\partial u}(u;y) - F'(u)]$ and by considering the adjoint equation, 12.35 reads:

$$\int_0^T \big\langle J'_{obs}(y^c), w^{\delta c}(t)\big\rangle_{V'\times V} dt = +\big\langle p(0), \delta y_0\big\rangle_H - \int_0^T \big\langle [\frac{\partial A}{\partial u}(u;y) - F'(u)]^T p(t), \delta u(t)\big\rangle_{\mathcal{U}'\times\mathcal{U}} dt \tag{12.37}$$

Next, by combining (12.34), (12.37) and (12.36), we obtain:

$$j'(c)\cdot\delta c \;=\; J'_{reg}(c)\cdot\delta c \;+\; \big\langle p(0), \delta y_0\big\rangle_H \;-\; \int_0^T \big\langle [\frac{\partial A}{\partial u}(u;y) - F'(u)]^T p(t), \delta u(t)\big\rangle_{\mathcal{U}'\times\mathcal{U}}\, dt \tag{12.38}$$

with (recall) $c(x,t) = (y_0(x), u(x,t))$ and $J'_{reg}(c)\cdot\delta c = \alpha_{reg,0}(J^0_{reg})'(y_0)\cdot\delta y^0 + \alpha_{reg,u}(J^u_{reg})'(u)\cdot\delta u$.

The expression of $j'(c)\cdot\delta c$ above does not depend anymore on $w^{\delta c}$: this was the goal.

Moreover, since:

$$j'(c)\cdot\delta c = \frac{\partial j}{\partial y_0}(c)\cdot\delta y_0 + \frac{\partial j}{\partial u}(c)\cdot\delta u \ , \tag{12.39}$$

the gradient expression (12.32) follows by identification. □

A few remarks

In addition to the remarks already made in the finite dimension linear case, see Section 12.2, let us notice that:

- By construction the adjoint model is linear, whatever if the direct model is linear or not (like in the steady-state case).

- If the differential operator $A(u;y)$ is linear and symmetric (in $y(x,t)$), the problem is self-adjoint (see Section 11.6.2 for details): the adjoint model differs from the direct one by the RHS only.

- By considering the expression of J_{reg} and J_{obs} as defined in Section 12.1.2, we obtain:

$$\left\{\begin{array}{rcl} \partial_{y_0}j(c) & = & p^c(0) \ + \ \alpha_{reg,0}\ B(y_0 - y_b) \\ \partial_u j(c) & = & - \ \left[\dfrac{\partial A}{\partial u}(u;y^c) - F'(u)\right]^T \ p^c(t) \ + \ \alpha_{reg,u}\ (J^u_{reg})'(u) \end{array}\right. \tag{12.40}$$

Moreover, the RHS $J'_{obs}(y^c(t))$ of the adjoint model equals the data misfit term which reads: $Z^T R^{-1}(Zy(t) - z^{obs}(t))$.

Exercice 12.5. *Consider the time-dependent case of your practical (linear case or not).*
a) Write the adjoint equations; both in the weak and the classical forms.
b) Write the gradient expression based on the adjoint method.

The 1st order optimality system

As previously (steady-state system in the previous chapter), we can define the optimality system. It is the set of equations characterizing the optimal solution:

- the state equation,
- the adjoint state equation,

- the first order necessary optimality condition (the gradient must vanishes).

12.4 Numerical resolution: the 4D-Var algorithm

The so-called 4D-var algorithm denotes the optimal control algorithm for unsteady PDEs systems (a-priori in 3 space-dimensions plus time therefore the 4D terminology).

The algorithm

The adjoint model is time dependent, reverse in time.
Since the minimisation is performed by using an iterative descent algorithm (eg the quasi-Newton method BFGS), the algorithm reads as indicated in Fig. (12.1).
Given a *first guess* $c^0 \in \mathbb{R}^{n+m}$, compute $c^m \in \mathbb{R}^{n+m}$ making diminish the cost function $j(c)$, $j(c) \in \mathbb{R}$. To do so, at each iteration:

1) compute the cost function $j(c)$ by solving the direct model from 0 to T,

2) compute the gradient $\nabla j(c) \in \mathbb{R}^{n+m}$ by solving the adjoint model from T to 0,

3) given the current iteration c^n, the cost function value $j(c^n)$ and the gradient value $\nabla j(c^n)$, compute a new iterationc^{n+1} such that:

$$j(c^{n+1}) < j(c^n)$$

*) Iterate until convergence.

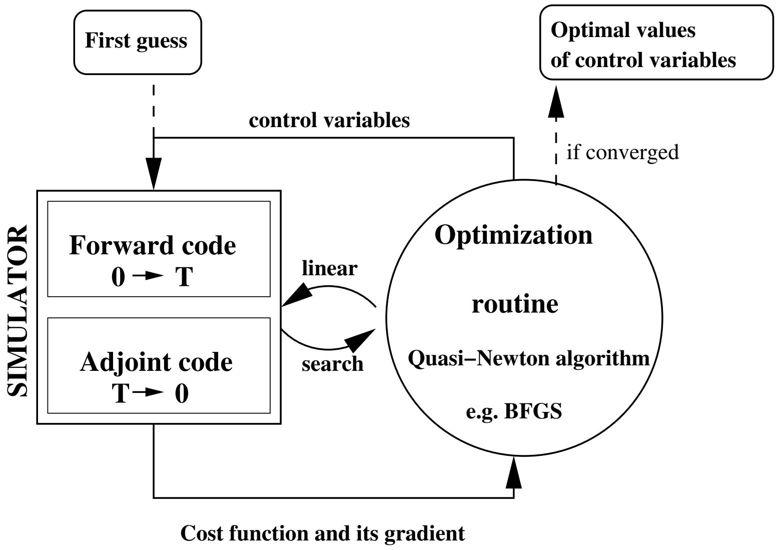


Figure 12.1: Optimal control algorithm for a time-dependent PDE model: the 4D-var algorithm.

A few remarks

Recall that the 4D-Var algorithm can be used for different goals.

A) To estimate the value of uncertain or unknown input parameters (time dependent or not), it is an *identification problem.*

B) To *calibrate* the model in order to perform better predictions; forecasting is the goal.
In this case, the data assimilation proceeds by *"analysis cycles"*. Figure 12.2 represents one cycle.

In this context,

-) the first stage is called the *"analysis step"*.
Observations of present and past time are assimilated to obtain the analysis i.e. the optimal state value (that is the optimal model trajectory).

-) the second stage consists to run the model in time: it is the *"forecasting step"*.

It is expected that if the model fits better the observations in the past, it will be more accurate for future.
Next, the forecast is used in the next analysis cycle etc

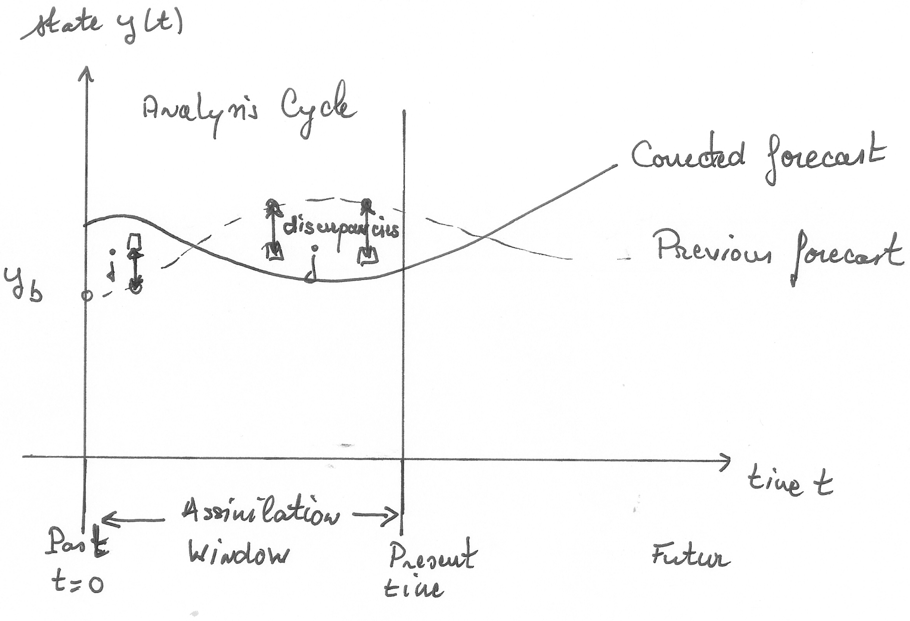


Figure 12.2: The 4D-var algorithm here used to identify the I.C. y_0. All observation within the assimilation time window are assimilated in the global least-square formulation (analysis step). Next, the resulting calibrated model is performed for prediction (forecasting step).

A concluding remark on hybrid approaches Data assimilation aims to fuse all available information in an “optimal way”. This includes the “physics” of the phenomena (e.g., conservation laws), the parameters (generally empirical), the initial condition (e.g., for weather forecasting), in-situ measurements, remote-sensed measurements (e.g., extracted from various satellite datasets), and prior probabilities (covariance operators defining the norms).
Additional measurements may be used to improve the “analysis”, especially if its confidence (accuracy) can be estimated by expertise or a statistical method.

Estimating uncertain parameters of physically-based models could be addressed by “non-physically informed” (“blind”) Machine Learning methods such as deep Neural Networks, if datasets are large enough.
A typical use of ML in this context would be to define the first guess by a NN. Next, the VDA process would act as a physically-informed filter.
In other respects, note that the present physically-informed approach enables the introduction of statistics and prior probabilities in the formulation.
Moreover, the VDA approach enables the assessment of the obtained estimations through the “physical model reading”.

12.5 The fundamental equations at a glance

The **time-dependent (non-linear) PDE model** reads:

$$(\mathcal{D})\left\{\begin{array}{l}\text{Given } (y_0(x), u(x,t)),\ \text{find } y(x,t) \text{ such that :}\\ \partial_t\, y(x,t) \ +\ A(u(x,t)\,;y(x,t)) \ =\ F(u(x,t)) \qquad \text{in } \Omega\times]0,T[\\ y(x,0) \ =\ y_0(x) \text{ in } \Omega\end{array}\right. \tag{12.41}$$

The direct model is supposed to be well-posed.
The control parameter is: $c(x,t) = (y_0(x), u(x,t))$.
Then, the control-to-state operator (the model operator) $\mathcal{M}(c)$ is defined as: $\mathcal{M}(c) = y^c(x,t)$.
$\mathcal{M}(\cdot)$ is a-priori non linear.

The **objective function** classically reads:

$$J(c\,;y) \ =\ \frac{1}{2}\int_0^T \left\| Z(y(t)) - z^{obs}(t)\right\|^2_{R^{-1}} dt \ +\ \alpha_0\frac{1}{2}\left\| y_0 - y_b\right\|^2_{B^{-1}} +\ \alpha_u J^u_{reg}(u) \tag{12.42}$$

In discrete form, the observations term $J_{obs}(c\,;y)$ reads: $\frac{1}{2}\sum_n \left(\sum_i Z(y^{obs}(x_i,t_n)) - z^{obs}_{i,n}\right)^2_{R^{-1}}$.

The **cost function** is defined as: $j(c) = J(c\,;y^c)$,
where y^c is the (unique) solution of the direct model given c.

The **optimization problem** reads:

$$\left\{\begin{array}{l}\text{Find } c^*(x,t) = (y_0^*(x), u^*(x,t)) \text{ such that:}\\ j(c^*) = \displaystyle\min_{c\in H\times\mathcal{U}_T} j(c)\end{array}\right. \tag{12.43}$$

The gradient components read: $\nabla j(c) = (\frac{\partial j}{\partial y_0}(c), \frac{\partial j}{\partial u}(c))^T$, with:

$$\left\{\begin{array}{rcl}\partial_{y_0} j(c(x,t)) &=& p^c(x,0) \ +\ \alpha_0\ B^{-1}(y_0 - y_b)(x)\\ \partial_u j(c(x,t)) &=& \left(-[\partial_u A(u;y^c)] + [F'(u)]\right)^T \cdot p^c(x,t) + \ \alpha_u\ (J^u_{reg})'(u(x,t))\end{array}\right. \tag{12.44}$$

where p^c is the (unique) solution of the **adjoint model**:

$$(\mathcal{A})\left\{\begin{array}{l}\text{Given } c = (y_0(x), u(x,t)) \text{ and } y^c(x,t) \text{ the unique solution of the direct problem } (\mathcal{D}),\\ \ \text{find } p(x,t) \text{ such that:}\\ -\partial_t\, p(x,t) + \Big[\partial_y A(u(x,t); y^c(x,t))\Big]^T p(x,t) = J'_{obs}(y^c(x,t)) \quad \text{in } \Omega\times]0,T[\\ p(x,T) \ =\ 0 \text{ in } \Omega\end{array}\right. \tag{12.45}$$

12.6 Complexity reduction & incremental 4D-Var algorithm*

This is a "to go further" section.

12.6.1 Basic principles

For very large scale problems (e.g. oceans and atmosphere in geophysics), it can be unaffordable to perform a 4D-var process as previously presented. In the case the assimilation is required for prediction (e.g. weather forecast), the forecast obviously needs to be performed faster than real time ! Even if based on mathematically reduced models e.g. the 2D shallow-water equations (reduced version of the complete 3D Navier-Stokes model with mobile surface), the 4D-var algorithm may remain too CPU time / memory consuming.

Then one can reduce the complete process by keeping the same (global) 4D-var control loop but by reducing the direct model following different methods.

- Develop a reduced basis like POD to solve the direct model, potentially combined with ML for non-linear models, see e.g. and [35] and references therein,
- Consider a much simpler physics instead of the original one (complet physics, fine grid) in the optimal control loop. If considering a linear simplified direct model (eg a linearized model at each iteration), this provides a LQ problem therefore much faster to solve. This is the basic idea of the so-called incremental 4D-var algorithm which is presented below.

12.6.2 Incremental 4D-var algorithm

The basic idea of the incremental 4D-var algorithm is to combine in the minimization process a low-resolution (or linearized equations) and the original full physics model.

Keeping the discrete notations previously introduced, the *innovation vector* d_n is defined by :

$$\boxed{d_n = (z_n - \mathcal{Z}(y_n)) \quad n \geq 0}$$

The innovation vector d_n measures at time t_n, the discrepancy between the model output and the observed quantity, in the observation space.

The idea is to modify the 4D-var algorithm as follows:

1) the iterative control variable corrections are performed using a low-resolution model (potentially both in physics and grids). This leads to low-resolution inner-loops.

2) Once these low-resolution inner-loops have converged, the discrepancy between the model and the measurements (i.e. the innovative vector) is computed by using the original complete direct model.

*) Repeat the process.

For a sake of simplicity, we define the operator model $\mathcal{M}$ as follows :

$$\boxed{y_n = \mathcal{M}(y_0)}$$

If we set: $y_b = y_0 + \delta y_0$, the sough quantity becomes δy_0 and we have:

$$j(y_0) = \frac{1}{2}\sum_{n=0}^{N}(\mathcal{Z}(y_n) - z_n)^T R^{-1}(\mathcal{Z}(y_n) - z_n) + \frac{1}{2}\delta y_0^T B^{-1}\delta y_0$$

Next, the "perturbation" δy_n corresponding to the perturbation δy_0 is defined as follows:

$$y_n + \delta y_n = \mathcal{M}(y_0 + \delta y_0)$$

A formal linearization (Taylor's expansion order 1) gives :

$$\mathcal{M}(y_0 + \delta y_0) \approx \mathcal{M}(y_0) + M(y_0)\cdot\delta y_0 \text{ hence } y_n + \delta y_n \approx y_n + M(y_0)\cdot\delta y_0$$

where M is the linear tangent model at time step t_n. Then : $\delta y_n \approx M(y_0)\cdot\delta y_0$.

Similarly:

$$\mathcal{Z}(y_n + \delta y_n) \approx \mathcal{Z}(y_n) + H_n\cdot\delta y_n$$

with H_n the linearized observation operator at time step t_n.

We have:

$$\boxed{j(y_0 + \delta y_0) \approx g(\delta y_0) = \frac{1}{2}\sum_{n=0}^{N}(H_n\cdot\delta y_n - d_n)^T R^{-1}(H_n\cdot\delta y_n - d_n) + \frac{1}{2}\delta y_0^T B^{-1}\delta y_0}$$

The basic idea of the incremental 4D-var method is to minimize the cost function with respect to the increment δy_0. Considering the cost function $g(\delta y_0)$, the inverse problem becomes *linear-quadratic optimal control problem* since $\delta y_n \approx M(y_0)\cdot\delta y_0$. Then the minimization can be performed using the Conjugate Gradient with preconditioning, hence very fast to compute.

In summary, the 4D-var incremental algorithm reads as follows (see Figure 12.3).

*) Inner loop.
The minimization process is performed with linearized operators (M and H_n), furthermore potentially with coarser grids and simplified physics.
Since the cost function $g(\delta y_0)$ is strictly convex (it is quadratic in δy_0), the extremely efficient

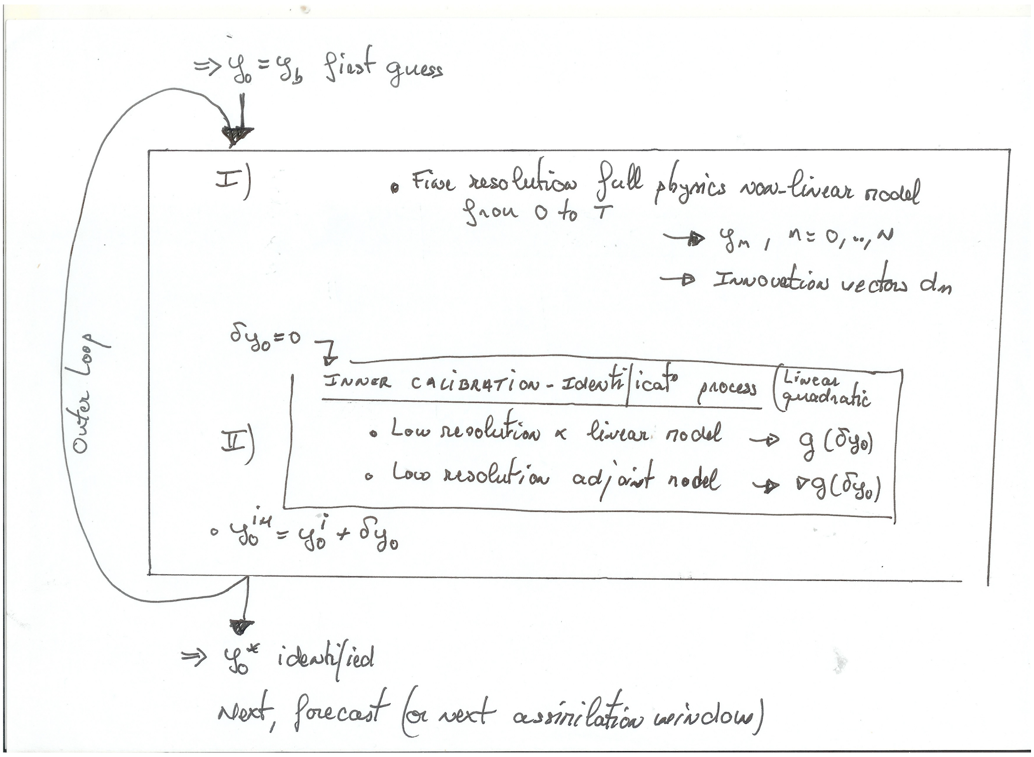

⇒ $y_0 = y_b$ first guess
Outer Loop
I)
• Fine resolution full physics non-linear model from 0 to T
→ y_n , $n = 0, .., N$
→ Innovation vector d_n
$\delta y_0 = 0$
INNER CALIBRATION - Identificat° process (Linear quadratic
II)
• Low resolution × linear model → $g(\delta y_0)$
• Low resolution adjoint model → $\nabla g(\delta y_0)$
• $y_0^{i+1} = y_0^i + \delta y_0$
⇒ y_0^* identified
Next, forecast (or next assimilation window)

Figure 12.3: The incremental VDA (4D-Var) algorithm. A simplified physical model in considered in the inner loop. However, the innovative is still computed using the complete direct model.

Conjugate Gradient method with preconditioning can be used to minimize g.

*) Outer loop.
Once the low-resolution minimization process has converged (the so-called "analysis step" is done), the original "high-resolution / full" model is performed (it is the so-called "prediction stage"). The prediction stage gives new innovation vectors (the discrepancy between the model and the observations).

Let us point out that the innovation vectors d_n are the crucial input of the assimilation method, thus they are computed using the high-resolution model.

*) Update the increment and the reference state.

Thus, *the incremental 4D-var method is a mix of calibration on linearized operators, full physic predictions and discrepancy measurements based on the fine innovation vector.*

For more details, the reader should consult [11, 40], where the method is developed in a weather forecast context.

The reader may consult [2] Chapter 5 too.

12.6.3 On hybrid approaches

Data assimilation aims at fusing in an ”optimal way” all available information: the ”physics” of the phenomena (e.g. conservation laws), the parameters (generally empirical), the initial condition (e.g. for weather forecast), the in-situ measurements, the remote-sensed measurements (e.g. extracted from various satellite datasets), prior probabilities (covariance operators defining the norms).

Additional measurements may be used to improve the ”analysis”, especially if its confidence (accuracy) can be estimated by expertise or a statistical method.

Estimating uncertain parameters of physical-based models could be addressed by ”non-physically informed” (”blind”) Machine Learning methods e.g. deep Neural Networks, if datasets are large enough.
A typical use of ML in the present context would be to define the first guess by a NN. Next, the VDA process would act as a physically-informed filter.

In other respect, note that the present physically-informed approach enables to introduce statistics and prior probabilities in the formulation.
Moreover, the VDA approach enables to assess the obtained estimations through the ”physical model reading”.

12.7 Exercises

The exercises below mainly consist to write the optimality system for classical PDE models. ADD exercices on line, lyx format, cf Moodle page.

12.7.1 Viscous Burgers' equation

The viscous Burgers' equation is the 1d simplification of the Navier-Stokes momentum equation. It is a scalar non-linear advection diffusion equation (non-linear advection term). The unknown is $u(x,t)$ the fluid velocity at point x and time t.
The control variables we consider in the present example are: the initial condition u_0 and the velocity value at one boundary extremity; the latter is denoted by v.
The forward (direct) model reads as follows.
Given $\mathbf{c} = (u_0, v)$, find u which satisfies:

$$\begin{cases} \partial_t u(x,t) - \nu\partial^2_{xx}u(x,t) + u\partial_x u(x,t) = f(x,t) & \text{in }]0,L[\times]0,T[\\ u(x,0) \ = \ u_0(x) & \text{in }]0,L[\\ u(0,t) = v(t) \ \ ; \ u(L,t) = 0 & \text{in }]0,T[\end{cases} \tag{12.46}$$

We assume we have m observations points of the flow, continuous in time. Then, we seek to minimize the following cost function:

$$j(\mathbf{c}) = \frac{1}{2}\int_0^T \sum_{i=1}^m |u(x_i) - u_i^{obs}|^2 dt$$

Exercice 12.6. *Write the optimality system corresponding to this data assimilation problem.*

12.7.2 Diffusion equation with non constant coefficients

We consider the diffusion equation (or heat equation) in an inhomogeneous media. Let u be the quantity diffused and $\lambda(x)$ be the diffusivity coefficient, non constant. The forward model we consider is as follows. Given λ and the flux φ, find u which satisfies:

$$\begin{cases} \partial_t u(x,t) - \partial_x(\lambda(x)\partial_x u(x,t)) = f(x,t) & \text{in } \Omega\times]0,T[\\ u(x,0) \ = \ u_0(x) & \text{in } \Omega \\ -(\lambda(x)\partial_n u(x,t) = \varphi & \text{in } \Gamma_1\times]0,T[\\ u(x,t) = 0 & \text{in } \Gamma_0\times]0,T[\end{cases} \tag{12.47}$$

with $\partial\Omega = \Gamma_0 \cup \Gamma_1$.
We assume we have measurements of the quantity u at boundary Γ_1, continuously in time. Then, we seek to minimize the following cost function:

$$j(\mathbf{c}) = \frac{1}{2}\int_0^T \int_{\partial\Omega} |u(x) - u^{obs}|^2 dsdt$$

Exercice 12.7. *Write the optimality system corresponding to this data assimilation problem.*

Part IV

Complements

Chapter 13

Code aspects

Validating a computational code is a mandatory step before performing simulations. In this chapter are presented two technics to validate the gradient computed by a code.
Moreover, the concept of twin experiment enabling to investigate the reliability of the VDA-based inversions is presented.

13.1 Validation of the computed gradients and adjoint codes

Below are described methods how to validate the adjoint code and the cost function gradient.

- *Validation of the adjoint code.* It can be verified that the code actually computes the adjoint of the tangent linear code by computing the scalar product property. This supposes however to have developed the tangent linear code too.
- *Validation of the gradient.* The adjoint-based gradient can be compared to finite differences values. This is the so-called the gradient test. This test should be done for any computational code computing a model output gradient.

If not interested today in practical computational aspects, this section may be skipped.

13.1.1 The scalar product test

This test aims at checking if the adjoint code is actually the adjoint of the Tangent Linear code. This test supposes to have developed both the adjoint code and the linear tangent code.

The test aims at numerically verifying the definition of an adjoint operator. Let M be a linear operator defined from $\mathcal{U}$ to $\mathcal{Y}$, we have:

$$< Mu, y >_{\mathcal{Y}} = < u, M^* y >_{\mathcal{U}}$$

Let u_0 be a given parameter value.

- Given an arbitrary perturbation $du \in \mathcal{U}$, the Tangent Linear code output is computed:

$$dy = \left(\frac{\partial \mathcal{M}}{\partial u}(u_0)\right) \cdot du$$

- Given an arbitrary perturbation $dy^* \in \mathcal{Y}$, the adjoint code output is computed:

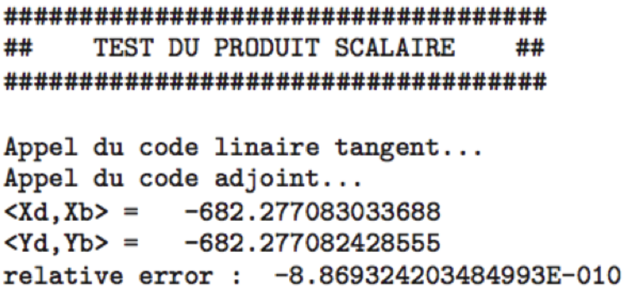
$$du^* = \left(\frac{\partial \mathcal{M}}{\partial u}(u_0)\right)^* \cdot dy^*$$

- The two following scalar products are computed:

$$sp_y = \left\langle dy^*, dy \right\rangle_{\mathcal{Y}} \text{ and } sp_u = \left\langle du^*, du \right\rangle_{\mathcal{U}}$$

- The validation relies on the relation: $sp_y = sp_u$.

Figure 13.1 (b) shows a typical example of the scalar product test.



```
###################################
##    TEST DU PRODUIT SCALAIRE    ##
###################################

Appel du code linaire tangent...
Appel du code adjoint...
<Xd,Xb> =    -682.277083033688
<Yd,Yb> =    -682.277082428555
relative error :   -8.869324203484993E-010
```

Figure 13.1: Adjoint code validation: scalar product test

13.1.2 The gradient test

The objective of this test aims at verifying that the gradient obtained from the adjoint corresponds to the partial derivatives of the cost function.
If first using an adjoint code, this test must be done before any further computations based on the adjoint-based gradient.
This test requires to perform a dozen of times the direct code and one time the adjoint code. The Tangent Linear code is here not required.

Let u_0 be a given parameter value. The Taylor expansion of the cost function j at u_0 for a small perturbation $\alpha\,\delta u$ ($\alpha \in \mathbb{R}^+$) reads:

$$j(u_0 + \alpha\,\delta u) \;=\; j(u_0) + \alpha\,\frac{\partial j}{\partial u}(u_0) \cdot \delta u + o\left(\alpha\|\delta u\|\right)\;. \qquad (13.1)$$

It follows the uncentered finite difference approximation (order 1) and the centered finite difference approximation (order 2):

$$\frac{j(u_0 + \alpha\,\delta u) - j(u_0 - \alpha\,\delta u)}{2\alpha} = \frac{\partial j}{\partial u}(u_0) \cdot \delta u + O\left(\alpha^2 \|\delta u\|^2\right) . \tag{13.2}$$

Then, we set either

$$I_\alpha = \frac{j(u_0 + \alpha\,\delta u) - j(u_0 - \alpha\,\delta u)}{2\alpha\,\frac{\partial j}{\partial u}(u_0) \cdot \delta u} \tag{13.3}$$

or

$$I_\alpha = \frac{j(u_0 + \alpha\,\delta u) - j(u_0)}{\alpha\,\frac{\partial j}{\partial u}(u_0) \cdot \delta u} \tag{13.4}$$

According to the Taylor expansions above, we have: $\lim\limits_{\alpha \to 0} I_\alpha = 1$.

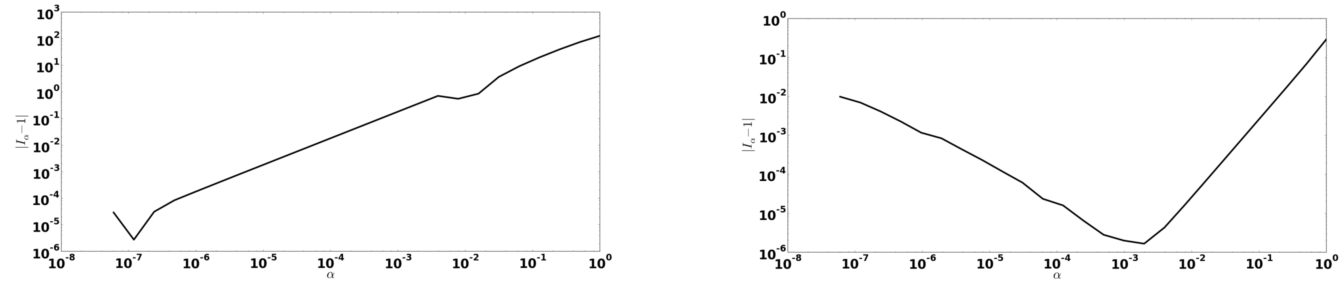
The gradient test consists to check this property as follows.

- Given an arbitrary parameter value u_0, compute $\frac{\partial j}{\partial u}(u_0)$ using the adjoint code.
- Using the direct code, compute $j(u_0)$.
- For $n = 0, \ldots, N$:
    - Compute $\alpha_n = 2^{-n}$;
    - Using the direct code, compute $j(u_0 + \alpha_n\,\delta u)$;
    - Compute I_{α_n} ;
- Verify if $\lim\limits_{\alpha \to 0} I_{\alpha_n} = 1$ or not.

Figure 13.2 shows two results of the gradient test: at order 2 and at order 1. $|I_\alpha - 1|$ is plotted vs α in logarithmic scale.
The convergence is good until $\alpha > 10^{-7}$.
However, observe the difference of accuracy between the 1st order and 2nd order approximation.

In the present exemple, the truncation errors errors appear for α smaller than $\approx 10^{-7}$ at order 1 ($\approx 10^{-3}$ at order 2). (To show this statement, one add a fix term in the Taylor expansion and notice that it is divided by the perturbation therefore increasing).

Exercice 13.1. *Perform the gradient test for your practical problem.*
Is your gradient computation valid ?



(a)

(b)

Figure 13.2: The adjoint code validation. Gradient test at order 1 (a), at order 2 (b).

13.2 Twin experiments

When addressing a real-world problem with a DA approach, the first mandatory step is to analyze *twin experiments* with increasing complexity. The principle of twin experiments is as follows.

- First, a dataset (the observations z^{obs}) is generated by applying the direct model to the input parameter u (this value will be referred to as the "true" value, denoted by u_t). The obtained observations are perfect in the sense that they are free from model errors. Next, noise (e.g., Gaussian noise with a realistic amplitude) may be added to these perfectly synthetic data.
- Second, the optimal control process is performed starting from an initial guess value u_b that differs from the "true" value u_t (the one used to generate the synthetic data).

As a consequence, since the true solution u_t corresponding to z^{obs} is known, thorough investigations can be conducted.

After the mandatory validation procedures (validations of the direct code, the adjoint code plus the gradient values), twin experiments are the next step to investigate the developed VDA formulation.

$$\begin{tikzcd}
\underset{\text{Input parameter}}{u_t} \arrow[r, "\text{Direct model}"'] & \underset{\text{Perfect Data}}{z_0^{obs}} \arrow[r] & \underset{\text{Noisy Data}}{z^{obs} = (z_0^{obs} + (\delta z)_{noise})}
\end{tikzcd}$$

Figure 13.3: Twin experiments. Step 1) Generation of synthetic data z^{obs}

$$\begin{tikzcd} z^{obs} \arrow[r] & u^* \\ \text{Synthetic data} & \text{Estimation } u^* \approx u_t? \end{tikzcd}$$

Figure 13.4: Twin experiments. Step 2) Inference of the input parameter u

Chapter 14

Regularization based on covariance operators*

** This is a "to go further" section.*

14.1 Introduction

Let us consider back the following general *non-linear stationary PDE model*:

$$\left\{\begin{array}{l} \text{Given } u(x), \text{ find } y(x) \text{ such that:} \\ A(u(x); y(x)) = F(u(x)) \text{ in } \Omega \\ \text{with Boundary Conditions on } \partial\Omega \end{array}\right. \tag{14.1}$$

The direct model above defines the control-to-state operator $\mathcal{M} : \; u \mapsto y^u$, y^u the unique solution given u.

The optimization problem reads:

$$\left\{\begin{array}{l} \text{Minimize } j(u) = J(u; y^u) \text{ in } U_{ad} \\ \;\text{with } \; y^u = \mathcal{M}(u). \end{array}\right. \tag{14.2}$$

where the observation function $J(u; y)$ is defined as, see (11.3):

$$J(u; y) \;=\; J_{obs}(y) + \alpha_{reg} J_{reg}(u) \tag{14.3}$$

$$\text{with } J_{obs}(y) = \|Zy - z^{obs}\|^2_{R^{-1}} \text{ and } J_{reg}(u) = \|u - u_b\|^2_{B^{-1}} \tag{14.4}$$

where u_b denotes the "background value", R^{-1} and B^{-1} denote symmetric (semi-)definite operators therefore (semi-)norms.

As already discussed in Section 2.2, without regularization term (equivalently $\alpha_{reg} = 0$), the inverse problem above can be ill-posed in the sense the optimal solution u^* can be non unique.

The computed optimal solution(s) u^* can depend on the first value $u^{(0)}$ of the iterative minimization process. In such cases, the choice of $u^{(0)}$ is crucial. $u^{(0)}$ may be determined from a good expertise of the modeled phenomena or from available data and priors.

Moreover, even if the problem is well-posed or $u^{(0)}$ well chosen, it is classical that the inverse problem is ill-conditioned: the cost function $j(u)$ is nearly flat in the vicinity of the minimum u^t, see Fig. 2.4.

This is for these two reasons that a regularization term $J_{reg}(u)$ may be introduced. As defined above, $J_{reg}(u)$ locally "convexify" the cost function in a vicinity of the background value u_b. In this case, the computed solution u^* depends on u_b and the weight parameter α_{reg} too.

The covariance matrix of observation errors R in $J_{obs}(y)$ should rely on a-priori statistical knowledge on the observations errors, see Section 5.1.2 and Section **??**. This is fully dependent on the modeled phenomena and the employed instruments. This point is not discussed in the present general context.

The error covariance matrix B in $J_{reg}(u)$ may rely on knowledge on the background errors which are generally badly known or even unknown... In the Bayesian context, this term corresponds to the prior $p(u)$ whose is assumed to be Gaussian, see Section **??**.

Another option is to define B from a-priori probabilistic model(s) or even from simplified physics of the modelled phenomena (see e.g. [**?**] for a spatial hydrology problem).

This chapter aims at showing:

1) how a natural change of variable is equivalent to pre-conditioning the optimality condition $\nabla J_{reg}(u) = 0$,

2) a few equivalences between classical covariance operators (Gaussian, second-order auto regressive kernel) and regularization terms,

3) links between classical covariance kernels and diffusive physical models.

14.2 Change of parameter variable, preconditioning

Let us consider the term $J_{reg}(u)$ with B symmetric positive definite. We define $B^{\frac{1}{2}}$ such that: $B = B^{\frac{1}{2}}B^{\frac{1}{2}}$. $B^{\frac{1}{2}}$ is symmetric positive definite.
We have:

$$J_{reg}(u) = \|u - u_b\|^2_{B^{-1}} = (B^{-1}(u - u_b), (u - u_b))_2 = \|B^{\frac{1}{2}}(u - u_b)\|^2_2 \qquad (14.5)$$

Then, it is quite natural to consider the change of variable $v = B^{\frac{1}{2}}(u - u_b)$ to obtain the simple expression: $J_{reg}(u) = \|v\|^2_2$.

However, the computation of $B^{-\frac{1}{2}}$ is CPU-time consuming. On the contrary the Cholesky decomposition of B^{-1}, $B^{-1} = L_B L_B^T$, is quite low CPU time-consuming.
By considering the change of variable:

$$v = L_B^T(u - u_b), \tag{14.6}$$

we obtain:

$$J_{reg}(u) = \|u - u_b\|_{B^{-1}}^2 = \|L_B^T(u - u_b)\|_2^2 = \|v\|_2^2 \equiv G_{reg}(v) \tag{14.7}$$

with $J_{reg} = G_{reg} \circ \mathcal{C}$, $\mathcal{C} : u \mapsto v = L_B^T(u - u_b)$.
Then, we have:

$$\boxed{\nabla G_{reg}(v) = L_B^{-T}\ \nabla J_{reg}(u)} \tag{14.8}$$

Therefore, the change of variable (14.6) modifies the descent directions during the minimisation process. Moreover, the change of variable may be perceived as a preconditioner of the first order necessary optimality condition $\nabla J_{reg}(u) = 0$. The introduction of B provides a different optimization "path"; hopefully more robust and faster convergence. This is the expectation...

The original parameter value u is simply recovered by applying the inverse of the (linear) change of variable as: $u = L_B^{-T}v + u_b$.

In practice, L_B is computed with few lower-diagonals only that is defining an incomplete Cholesky decomposition only.

14.3 Equivalences between B^{-1}-norms and regularization terms

The norm $\|\cdot\|_{B^{-1}}$ in the definition of $J_{reg}(u)$ is defined from a symmetric, positive linear operator (a matrix in finite dimension) therefore a covariance operator. The most commonly employed covariance operators are likely the following two, see e.g. [?] Chapter III and references therein.

ToDo: A POURSUIVRE: cf .tex

Chapter 15

Algorithmic - Automatic Differentiation*

** This is a ”to go further” section.*

Bibliography

[1] Marc Allaire and Alexandre Ern. *Optimisation et Contrôle.* Lecture notes, Ecole Polytechnique, 2024.

[2] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications.* SIAM, 2016.

[3] Mark Asch, Marc Bocquet, and Maëlle Nodet. *Data assimilation: methods, algorithms, and applications.* SIAM, 2016.

[4] Richard C Aster, Brian Borchers, and Clifford H Thurber. *Parameter estimation and inverse problems.* Elsevier, 2018.

[5] Combettes P.-L. Bauschke H. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces.* Springer, 2011.

[6] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[7] Leonard David Berkovitz and Negash G Medhin. *Nonlinear optimal control theory.* CRC press, 2012.

[8] Marc Bocquet. *Introduction to the principles and methods of data assimilation in the geosciences.* Lecture notes, Ecole des Ponts ParisTech.

[9] M. Bonavita. Overview of data assimilation methods. ECMWF course online, 2019.

[10] Joseph-Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia A Sagastizábal. *Numerical optimization: theoretical and practical aspects.* Springer Science & Business Media, 2006.

[11] L. Bouttier and P. Courtier. *Data assimilation concepts and methods.* ECMWF Training course. www.ecmwf.int., 1999.

[12] H. Brézis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations.* Springer, 2010.

[13] Edoardo Calvello, Sebastian Reich, and Andrew M Stuart. Ensemble kalman methods: A mean field perspective. *arXiv preprint arXiv:2209.11371*, 2022.

[14] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[15] Guy Chavent. *Nonlinear least squares for inverse problems: theoretical foundations and step-by-step guide for applications.* Springer Science & Business Media, 2010.

[16] Sibo Cheng, César Quilodrán-Casas, Said Ouala, Alban Farchi, Che Liu, Pierre Tandeo, Ronan Fablet, Didier Lucor, Bertrand Iooss, Julien Brajard, et al. Machine learning with data assimilation and uncertainty quantification for dynamical systems: a review. *IEEE/CAA Journal of Automatica Sinica*, 10(6):1361–1387, 2023.

[17] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[18] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[19] Lawrence C. Evans. *An Introduction to Mathematical Optimal Control Theory.* University of California, Berkeley, 2014.

[20] Geir Evensen. *Data assimilation: the ensemble Kalman filter.* Springer Science & Business Media, 2009.

[21] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis.* Chapman and Hall/CRC, 1995.

[22] David Gilbarg, Neil S Trudinger, David Gilbarg, and NS Trudinger. *Elliptic partial differential equations of second order*, volume 224. Springer, 1977.

[23] Per Christian Hansen. *Discrete inverse problems: insight and algorithms.* SIAM, 2010.

[24] Kayo Ide, Philippe Courtier, Michael Ghil, and Andrew C Lorenc. Unified notation for data assimilation: Operational, sequential and variational (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):181–189, 1997.

[25] Jari Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*, volume 160. Springer Science & Business Media, 2006.

[26] Barbara Kaltenbacher, Andreas Neubauer, and Otmar Scherzer. *Iterative regularization methods for nonlinear ill-posed problems*, volume 6. Walter de Gruyter, 2008.

[27] Michel Kern. *Numerical methods for inverse problems.* John Wiley & Sons, 2016.

[28] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Andreas Kirsch. *An introduction to the mathematical theory of inverse problems*, volume 120. Springer Science & Business Media, 2011.

[30] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[31] J.L. Lions. *Contrôle optimal de systèmes gouvernés par des équations aux dérivées partielles*. Dunod, 1968.

[32] J.L. Lions. *Optimal control of systems governed by partial differential equations*. Springer-Verlag, 1971.

[33] JL Lions and R Dautray. Evolution problems i. mathematical analysis and numerical methods for science and technology, vol. 5, 2000.

[34] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[35] J. Monnier. *Finite Element Methods & Model Reductions*. INSA - University of Toulouse. Open Online Course, 2022.

[36] Jennifer L Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[37] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[38] Albert Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. siam, 2005.

[39] E. Trélat. *Optimal control: theory and applications*. Vuibert, Paris, 2008.

[40] Y. Trémolet. Incremental 4d-var convergence study. *Tellus A*, 59(5):706–718, 2008.

[41] Fredi Tröltzsch. *Optimal control of partial differential equations: theory, methods, and applications*, volume 112. American Mathematical Soc., 2010.

[42] Curtis R Vogel. *Computational methods for inverse problems*. SIAM, 2002.

Appendices

Appendix A

Basic recalls of Differential Calculus and Optimization

Please consult the supplementary material.